UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Politècnica Superior d'Enginyeria de Manresa

Treball Final de Grau

Desenvolupament d'un programari pel disseny, resolució i correcció automàtica de problemes d'enginyeria química

Grau en Enginyeria química

Curs 15/16

Autor: Carles Buxaderas Vivancos

Director: Antonio David Dorado Castaño

Data: 14 d'Octubre de 2016

Localitat: Manresa (EPSEM, UPC)

RESUM DEL PROJECTE

En aquest projecte s'ha desenvolupat un programari utilitzant Matlab i ThingSpeak que permet dissenyar, resoldre i corregir de forma automàtica problemes d'enginyeria química, així com una guia per a dissenyar els problemes utilitzant el programari com a model base.

Gràcies a l'ampli ventall de possibilitats que ofereix Matlab, el programari pot ser utilitzat amb gran diversitat de problemes, i tot i que en un inici el programari ha sigut desenvolupat pel seu ús en l'àmbit docent, la seva versatilitat permet adaptar-lo per a multitud de funcions de l'entorn industrial com tasques de disseny, operació i optimització.

La característica que fa únic al programari de cara a la seva aplicació en l'àmbit docent és la capacitat de tractar les dades generades pels alumnes amb el seu ús, permetent al docent disposar de recursos per interpretar i analitzar les dades i automatitzar la cerca d'errors comuns entre els usuaris.

Anteriorment al desenvolupament del programari , s'ha realitzat un estudi de l'entorn de la seva aplicació, analitzant el paper de l'avaluació en l'àmbit docent i investigant mètodes d'aprenentatge moderns que en potenciïn el seu ús, així com de treballs previs en la mateixa línia.

PROJECT ABSTRACT

Through this project a software has been developed using Matlab and ThingSpeak. This software allows to solve and automatically correct chemical engineering problems. Also a guide has been developed for the design of problems using the software as a base model.

Thanks to the wide range of possibilities offered by Matlab, the software can be used in a wide variety of problems, and although initially the software has been developed for teaching purposes, its versatility allows it to adapt to a multitude of functions and tasks for the industrial environment, such as design, operation and optimization.

The feature that makes the software unique for the teaching environment is its ability to process data generated by the students on its use, allowing the teacher to have resources for the interpretation and analysis of data and to automate the search of common mistakes among users.

Previously to the development of the software, a study of previous works on the field has been made and the role of evaluation and modern learning methods have been analyzed in order to search for ideas to boost the software and study its environment of application.

ÍNDEX

1. Introducció i objectius 9
2. Avaluació i aprenentatge basat en problemes (ABP) 11
3. Antecedents 15
4. Programari desenvolupat 17
4.1 Introducció al programari desenvolupat 17
4.2 Problema exemple 19
4.3 Diagrama de interacció amb l'usuari 21
4.4 Interfície i interacció amb l'usuari 22
4.5 Diagrama de funcionament del codi 27
4.6 Conceptes previs de Matlab, GUIDE i ThingSpeak 28
4.6.1 Comunicació del codi amb la interfície 28
4.6.2 Càlculs 30
4.6.3 Adequació de format 31
4.6.4 Comprovacions 33
4.6.5 Condicionals 34
4.6.6 Bucles 34
4.6.7 Anotacions 35
4.6.8 ThingSpeak 35
4.7 Taula de variables 37
4.8 Identificació del codi corresponent a cada funció i codi en detall 42
4.8.1 Funcions que no participen en la resolució del problema 42
4.8.2 Funció "Funció obertura" 44
4.8.3 Funció "Genera paràmetres" 45
4.8.4 Funció "Avalua " 49
4.8.5 Funció "Envia resultats" 57
4.9 ThingSpeak. Emmagatzematge de resultats i tractament de dades 61
4.9.1 Canals 61
4.9.2 Exportació de dades 63
4.9.3 Tractament de dades 64
5. Disseny d'un problema 77
5.1 Problema exemple 2 77
5.2 Interfície 84

5.3 Codi 89
5.3.1 Funció "Funció obertura" 90
5.3.2 Funció "Genera paràmetres" 90
5.3.3 Funció "Avalua" 93
5.3.4 Canals de ThingSpeak 100
5.3.5 Funció "Envia resultats" 106
5.3.6 Fitxer distribuïble de codi ocult 110
5.4 Visualitzacions 111
5.5 Anàlisi de dades: segona avaluació amb un error identificat 117
5.6 Exportació de dades 122
6. Tipologia de problemes 126
7. Conclusions 127
8. Futures línies de treball 128
9. Bibliografia 129

ÍNDEX DE FIGURES

Figura 1: Logo de Matlab. Font: https://engineering.usu.edu/students/matlab 17
Figura 2: Logo de ThingSpeak. Font: https://thingspeak.com/ 17
Figura 3: Representació gràfica del procés. Problema exemple. Font: Sistemes Químics. Llistat de problemes finals, 1. 19
Figura 4: Diagrama de interacció amb l'usuari. Font: Pròpia. 21
Figura 5: Aspecte inicial de la interfície. Font: Pròpia 22
Figura 7: Errors al generar paràmetres. Font: Pròpia 23
Figura 6: Introducció de DNI i botó "Genera paràmetres". Font: Pròpia 23
Figura 8: Paràmetres generats. Font: Pròpia 24
Figura 9: Introducció de respostes. Font: Pròpia 24
Figura 10: Nota i comentaris. Font: Pròpia 25
Figura 11: Nota i comentaris. Font: Pròpia 26
Figura 12: Diagrama de funcionament del codi. Font: Pròpia 27
Figura 13: Taula d'error en segona avaluació. Font: Pròpia 32
Figura 14: Canal 1 problema exemple. Font: Pròpia. 61
Figura 15: Canal 2 problema exemple. Font: Pròpia. 62
Figura 16: Canal 3 problema exemple. Font: Pròpia. 62
Figura 17: ID I claus canal 1 problema exemple. Font: Pròpia. 63
Figura 18: Opcions de tractament de dades. Font: https://thingspeak.com/apps 64
Figura 19: Histograma de notes. Font: Pròpia. 68
Figura 20: Histograma d'errors. Font: Pròpia. 71
Figura 21: Esquema del sistema. Qüestió 1 problema exemple 2. Font: Pròpia. 78
Figura 22: Esquema del sistema. Qüestió 2 problema exemple 2. Font: Pròpia. 78
Figura 23: Accés a GUIDE. Font: Matlab. 84
Figura 24: Creació d'interfície en blanc. Font: Matlab. 84
Figura 25: Opcions de la barra d'eines emprades. Font: GUIDE, Matlab. 85
Figura 26: Distribució de caselles de text no editable. Font: Pròpia. 86
Figura 27: Panell de propietats d'una casella de text no editable. Font: Pròpia. 86
Figura 28: Caselles de text no editables amb format ajustat i "tags". Font: Pròpia. 87
Figura 29: Caselles de text editables i "tags". Font: Pròpia. 88
Figura 30: Botons i "tags". Font: Pròpia. 89
Figura 31: Portada de ThingSpeak. Font: https://thingspeak.com/ 100
Figura 32: Creació de compte de ThingSpeak. Font: https://thingspeak.com/ 101
Figura 33: Panell de creació de canal. Font: https://thingspeak.com/. 101

Figura 34: Nom i descripció del canal. Font: Pròpia. 102
Figura 35: Camps del canal 1 problema exemple 2. Font: Pròpia. 102
Figura 36: Casella per convertir el canal en públic. Font: https://thingspeak.com/..... 103
Figura 37: Vista privada del canal. Font: https://thingspeak.com/..... 103
Figura 38: Accés al llistat de canals. Font: https://thingspeak.com/..... 104
Figura 39: Llistat de canals. Font: https://thingspeak.com/..... 105
Figura 40: Camps del canal 2 problema exemple 2: Font: Pròpia. 105
Figura 41: Camps del canal 3 problema exemple 2. Font: Própia. 106
Figura 42: Accés a aplicacions de ThingSpeak. Font: https://thingspeak.com/..... 111
Figura 43: Accés a les visualitzacions. Font: https://thingspeak.com/. 111
Figura 44: Panell de creació de visualitzacions. Font: https://thingspeak.com/..... 112
Figura 45: Opció de plantilla de codi per visualització. Font: https://thingspeak.com/. 112
Figura 46: Pàgina de disseny de visualització. Font: https://thingspeak.com/. 113
Figura 47: Accés a les anàlisis. Font: https://thingspeak.com/. 117
Figura 48: Pàgina de disseny d'anàlisi. Font: https://thingspeak.com/..... 118
Figura 49: Taula del resultat de l'anàlisi problema exemple 2. Font: Pròpia. 121
Figura 50: Casella d'exportació de dades. Font: https://thingspeak.com/. 122
Figura 51: Aspecte inicial de les dades descarregades. Font: https://thingspeak.com/. 122
Figura 52: Adequació de format de les dades descarregades (1). Font: Pròpia. 123
Figura 53: Adequació de format de les dades descarregades (2). Font: Pròpia. 123
Figura 54: Adequació de format de les dades descarregades (3). Font: Pròpia. 124
Figura 55: Adequació de format de les dades descarregades (4). Font: Pròpia. 124
Figura 56: Format final de les dades descarregades. Font: Pròpia. 125

ÍNDEX DE TAULES

Taula 1: Variables "Funció inicial" i "Genera paràmetres". Font: Pròpia 37
Taula 2:Variables "Avalua" 1.0 . Font: Pròpia..... 38
Taula 3: Variables "Avalua" 2.0 . Font: Pròpia..... 39
Taula 4: Variables "Envia resultats" 1.0. Font: Pròpia 40
Taula 5: Variables "Envia resultats" 2.0. Font: Pròpia 41
Taula 6: Dades exportades canal 1 problema exemple. Font: Pròpia. 63
Taula 7: Dades exportades canal 2 problema exemple. Font: Pròpia. 64
Taula 8: Correspondència entre el número d'identificació en l'eix horitzontal de l'histograma i la nota. Font: Pròpia. 66
Taula 9: Correspondència entre el número d'identificació en l'eix horitzontal de l'histograma amb l'error. Font: Pròpia. 70
Taula 10: Taula amb resultats de l'anàlisi de dades del problema 1. Font: Pròpia. 76
Taula 11: Paràmatres problema exemple 2. Font: Pròpia. 79
Taula 12: Valors extrems dels paràmatres en el problema exemple 2. Font: Pròpia. .. 83
Taula 13: Casos de combinacions d'extrems en el problema exemple 2. Font: Pròpia. 83
Taula 14: Taula d'equivalències d'errors problema exemple 2. Font: Pròpia. 94

1. Introducció i objectius

L'àmbit de la docència evoluciona de forma constant, adaptant-se a múltiples realitats i necessitats que hi col·lisionen.

En els últims anys, fruit de la convergència cap a l'Espai Europeu d'Educació Superior (EEES), els mètodes de docència d'aprenentatge basat en problemes (ABP) han esdevingut una corrent que ha agafat força gràcies als resultats positius que obtenen els centres que decideixen aplicar-los.

Un factor que dona suport a aquests mètodes i que es troba més present que mai en l'àmbit de la docència és la gran varietat d'eines informàtiques que pretenen facilitar les tasques tant d'alumnes com docents.

Actualment existeixen programes que busquen canviar antigues dinàmiques d'ensenyament amb les seves funcionalitats, però en cap s'ha desenvolupat la capacitat de tractar les dades generades pels usuaris en la interacció amb el programa.

Partint de l'interès despertat pels beneficis tant de l'ABP com de la incorporació d'eines informàtiques, s'estableix com a objectiu d'aquest projecte desenvolupar un programari que permeti introduir a les aules l'autoavaluació d'exercicis personalitzats amb correcció automàtica i feedback sobre els errors comesos i que, incloent aquesta capacitat única de tractar les dades generades, atorgui recursos al docent per interpretar i analitzar les dades recollides en la interacció dels usuaris amb el programa i permetent automatitzar la cerca d'errors comuns entre els usuaris.

Entre els beneficis que es cerquen amb l'ús d'un programari que inclogui aquestes funcionalitats es destaquen:

- Redueix el temps emprat en tasques de correcció automatitzades.
- Facilita l'avaluació continua i n'evita el plagi a partir d'exercicis personalitzats.
- Redueix subjectivitat en la correcció.
- Permet fer un seguiment més detallat del estat dels alumnes, ajudant a ajustar la dificultat de forma més dinàmica.
- Aporta informació a l'alumne sobre l'origen dels errors.

Es pretén també que a més de la seva aplicació a nivell docent, el programari serveixi de base per a diverses aplicacions que se'n puguin derivar en l'àmbit industrial per a tasques de disseny, operació o optimització.

2. Avaluació i aprenentatge basat en problemes (ABP)

Hi ha molts sistemes per avaluar, depenent del context i dels objectius docents.

En aquest apartat es busca reflexionar sobre el paper de l'avaluació en l'aprenentatge, introduint alhora la metodologia d'aprenentatge basat en problemes.

Qüestionar-se els motius de l'avaluació pot semblar una tasca fútil. És evident que l'avaluació és una part indispensable del procés d'aprenentatge. Però al pregunta'ns-ho emergeixen dimensions del procés d'avaluació sobre les que no es pensa gaire sovint (Equipo Docente en ABP. Facultad de Psicología. Universidad de Murcia. 2011).

Per tal de realitzar un anàlisi sobre l'avaluació i l'ABP es plantegen quatre preguntes: Què avaluar, qui avalua, quan avaluar i com avaluar.

- **Què avaluar?**

  Es pot partir de molt bones declaracions d'intencions per valors, actituds i competències que s'espera que l'alumne absorbeixi, però si l'obtenció d'una bona qualificació no és contingent a les declaracions difícilment seran adquirides pels estudiants.

  El que realment prengui pes en l'avaluació és el que l'alumne valorarà com a important. L'avaluació condiciona de tal manera la dinàmica de l'aula que es pot considerar que l'hora de la veritat no es la de l'aprenentatge, sinó la de l'avaluació.

  En l'ABP es considera que una avaluació autèntica s'ha de realitzar d'acord amb les demandes del món real, buscant que els alumnes treballin sobre problemes complexes mitjançant el pensament crític, sintetitzant informació diversa i buscant solucions originals. D'aquesta manera, s'aconsegueix una avaluació més idònia per a preparar als estudiants per les necessitats del món real.

L'avaluació de coneixements en el marc de l'ABP no es limita a una avaluació única, estandarditzada en una prova objectiva amb preguntes referenciades a coneixements discrets i descontextualitzats. L'objectiu d'un procés d'ABP ha de ser que l'estudiant aconsegueixi la comprensió profunda dels fenòmens i la interrelació de les disciplines.

En conseqüència, es busca utilitzar eines d'avaluació de coneixements coherents amb aquesta perspectiva. Per exemple, elaborar un mapa conceptual estaria en més sintonia amb el mètode que no pas memoritzar definicions.

A més, l'enfocament de l'ABP implica que, a banda d'avaluar coneixements, s'ha de donar pes a les habilitats i capacitats (recerca d'informació, expressió oral, definir un problema...), així com a les actituds i valors (promoure l'harmonia en el grup, respectar una ètica professional....)

- **Qui ha d'avaluar?**

Entenent l'avaluació com una puntuació que valora el rendiment de l'alumne, correspon al professor la responsabilitat principal de l'avaluació.

No obstant, incloure a l'alumne en el procés d'avaluació l'involucra en un nivell més profund en el procés d'aprenentatge i permet que l'alumne desenvolupi uns criteris de qualitat, reflexioni sobre els seus procediments i busqui millorar-los, creixent en autonomia i corresponsabilitat per tal d'assimilar unes bases per l'aprenentatge al llarg de la vida.

Una eina que resulta útil per involucrar a l'alumne en el procés d'avaluació i fomentar alhora la confiança i la sinceritat és l'avaluació negociada, que consisteix en proposar una nota a l'alumne per la seva feina i demanar-li la opinió sobre aquesta, fomentant també la reflexió del seu propi procés per decidir si la nota li sembla adequada al seu desenvolupament.

Un altre mètode per incorporar a l'alumne en el procés d'avaluació és l'autoavaluació. L'autoavaluació permet reflexionar a l'alumne sobre la seva trajectòria i considerar si esta complint els objectius, a més d'aportar consciència sobre els punts dèbils.

- **Quan avaluar?**

Avaluar serveix per comprovar l'assoliment d'objectius. Permet conèixer als estudiants com s'estan acostant als objectius plantejats i jutjar l'estat dels seus coneixements. A més, l'avaluació esdevé una eina útil per a la consecució d'objectius. És important doncs tenir en compte el moment de l'avaluació per tal de maximitzar els beneficis del feedback generat i per tal d'evitar que el procés d'avaluació esdevingui una simple sentencia.

En l'ABP l'avaluació es pren com un procés continu i omnipresent. Òbviament partint d'aquesta definició es deriven molts problemes, ja que avaluar-ho tot en tot moment de la forma més analítica possible resulta una tasca impossible. Però partint d'aquesta idea i entenent les limitacions, es centra l'objectiu en buscar la millor forma d'assolir les competències i no tant en trobar la mesura perfecta i exacta de l'avaluació.

- **Com avaluar?**

En l'ABP el treball en grup pren un pes important, ja que és un marc de treball més pròxim a les condicions del món real i permet treballar competències en les quals és impossible aprofundir de forma individual. Tot i així, el pes que es dona a la nota individual dependrà del context de l'assignatura i les competències a avaluar.

Donant tanta importància al paper de l'avaluació es corre el risc de patir una sobrecarrega avaluativa, tant pel professor com per l'alumne. Es pot passar d'un sistema tradicional molt més anònim a un sistema sufocant on s'avalua absolutament tot.

Això pot comportar problemes a nivell de grup, generant un ambient artificial on es doni un comportament políticament correcte forçat que, lluny de potenciar l'interès de l'alumne, generi estrès i saturació.

Per tal de fer arrels en els alumnes és important potenciar el punt d'interès i motivació, ja que en la seva absència l'atenció es veu minvada, així com l'eficiència del procés cognitiu.

Separar els termes avaluar i puntuar pot ser una solució per sortir d'aquesta situació. Amb aquesta perspectiva, el rol del docent és més semblant al d'un entrenador que al del clàssic professor, assenyalant punts forts i dèbils d'una forma idealment continua.

Seguint el mateix exemple de l'entrenador, el rol del docent queda més desplaçat a estimular, proposar, plantejar, detectar i superar limitacions i posar en joc les capacitats. El docent perd l'objectivitat del pur analista per canviar la situació amb la seva intervenció, sempre amb el propòsit de millorar la situació.

Així doncs, el professor anota, corregeix i retorna als estudiants individualment i en grup les seves observacions mentre els propis implicats s'autoavaluen i s'avaluen mútuament. És important però deixar un temps per digerir el feedback, per tal de permetre als estudiants explorar sense sentir-se pressionats.

D'aquesta manera s'aconsegueix que l'avaluació prengui més valor formatiu, és a dir, que es millori el que està sent avaluant en tot el possible.

En conclusió, la relació entre una avaluació innovadora i l'ABP és recíproca: l'ABP fa possible un altre forma d'avaluació al mateix temps que ho exigeix.

3. Antecedents

Partint de la filosofia de l'ABP i de l'objectiu que es busca amb aquest projecte, s'ha realitzat una recerca d'altres projectes que caminin en la mateixa línia.

En l'àmbit de generació de problemes amb correcció automàtica, existeix un projecte presentat en el segon congrés d'innovació docent en enginyeria química a València realitzat al gener de 2014 anomenat Goodle-GMS, creat pels professors Fabio Gómez-Stern i David Muñoz del Departament d'Enginyeria de Sistemes i Automàtica de la Universitat de Sevilla.

Després d'establir contacte per correu amb Carlos Leiva, un professor col·laborador del projecte Goodle-GMS, se'ns és facilitada una clau de prova i material didàctic per tal d'investigar el que ofereix.

Durant la realització d'aquest projecte, Goodle-GMS ha estat actualitzat amb més funcionalitats, canviant el seu nom a Doctus.

Doctus és una plataforma web que permet:

- Gestionar grups d'alumnes: subscripcions, assistència i entregues.
- Avaluació d'exercicis amb Matlab, C, C++ i Excel.
- Generar anunciats personalitzats.

Amb les funcions que ofereix resulta una eina pràctica per a l'avaluació continua, aportant feedback en temps real a l'alumne i estalviant temps en tasques de correcció mecàniques.

La seva utilitat està ben demostrada, augmentant el número d'aprovats i de tutories, a més d'estimular la formació de grups per resoldre problemes en les assignatures que s'ha implementat.

Per contra, Doctus no incorpora la possibilitat d'emmagatzemar les dades generades més enllà de la nota de l'alumne, cosa que fa impossible l'objectiu que es busca assolir amb aquest projecte: atorgar recursos al docent per interpretar i analitzar les dades recollides en la interacció dels usuaris amb el programa i automatitzar la cerca d'errors comuns entre els usuaris.

A més, Doctus és una plataforma tancada limitada exclusivament a l'àmbit docent, cosa que redueix el seu potencial per a altres possibles aplicacions.

4. Programari desenvolupat

4.1 Introducció al programari desenvolupat

El programari desenvolupat ha estat dissenyat amb dues eines principals: Matlab i ThingSpeak.

Matlab és un entorn de càlcul tècnic amb prestacions pel càlcul numèric amb el qual s'ha dissenyat el codi i la interfície.

Figura 1: Logo de Matlab. Font: https://engineering.usu.edu/students/matlab

La interfície en concret ha estat dissenyada amb GUIDE (graphical user interface development enviornment). GUIDE és una eina de Matlab per crear aplicacions de software que permeten automatitzar tasques i càlculs.

Mitjançant Matlab es genera un arxiu de codi ocult que conté el codi i la interfície corresponents a un problema que es distribueix entre els usuaris i que s'executa també amb Matlab.

Per altra banda ThingSpeak és una plataforma de dades oberta que brinda serveis destinats a construir gran varietat d'aplicacions, permetent l'emmagatzematge de dades i el seu tractament amb diverses possibilitats d'anàlisi, monitoratge i opcions d'acció i reacció.

Figura 2: Logo de ThingSpeak. Font: https://thingspeak.com/

El programari doncs, queda dividit en dos blocs: l'arxiu que es distribueix entre els usuaris i la plataforma que permet gestionar les dades generades amb l'ús de l'arxiu.

En executar l'arxiu l'usuari es troba amb una interfície interactiva on s'hi planteja un problema que ofereix les opcions de:

- Obtenir paràmetres personalitzats pel problema a partir del DNI de l'usuari.
- Correcció automàtica i generació de feedback per l'usuari. El feedback inclou la nota i comentaris sobre els errors comesos.
- Enviament de les dades generades amb les interaccions dels usuaris amb el programa a ThingSpeak.

Per altra banda, ThingSpeak permet al docent que distribueix l'arxiu amb el problema tractar les dades emmagatzemades.

És aquesta capacitat de tractar les dades emmagatzemades que aporta un factor més innovador al programari, ja que permet:

- Generar visualitzacions que faciliten el seguiment general del problema (com histogrames de notes i errors comesos).
- Automatitzar una segona avaluació de les dades. Aquesta funció és especialment útil un cop identificat un error categoritzat com a desconegut pel programa al analitzar les resolucions dels usuaris, ja que permet comprovar quants dels errors desconeguts són del mateix tipus que l'error identificat.

En els següents apartats es troben les explicacions per comprendre el funcionament del programa desenvolupat tant a nivell d'usuari com a nivell de codi i funcionament, així com de ThingSpeak i les seves opcions d'emmagatzematge i tractament de dades.

L'explicació comença amb el plantejament del problema exemple que s'utilitza com a model i procedeix de general a específic de manera que primerament s'explica el programa a nivell d'usuari i seguidament s'expliquen nocions bàsiques de Matlab per tal de comprendre el codi i les diferents funcions del qual està composat, així com el seu funcionament fins a l'etapa final d'emmagatzematge i tractament de dades.

4.2 Problema exemple

El problema exemple que s'utilitza com a model es tracta d'un problema senzill de balanços de matèria de l'assignatura de sistemes químics, essent el seu enunciat el que es mostra a continuació.

Enunciat problema exemple:

- Es pretén desalinitzar aigua de mar mitjançant osmosi inversa utilitzant el següent procés:

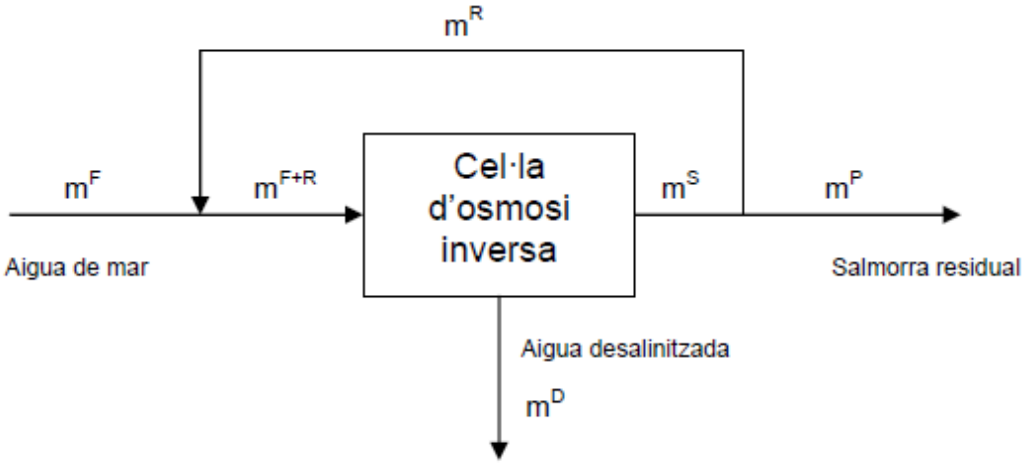

Figura 3: Representació gràfica del procés. Problema exemple. Font: Sistemes Químics. Llistat de problemes finals, 1.

Determinar:

a) El cabal de salmorra residual (m^P, kg/h)
b) La velocitat de producció d'aigua desalinitzada (m^D, kg/h)
c) La quantitat de salmorra que entra a la cel·la d'osmosi per a ser tractada (m^{F+R}, kg/h)

Per resoldre el problema es proporcionen els següents paràmetres generats a partir del DNI de l'usuari :

- m^F (cabal màssic d'aliment)
- w^Fsal (fracció màssica de sal en l'aliment)
- w^{F+R}sal (fracció màssica de sal en l'entrada de la cel·la)
- w^Psal (fracció màssica de sal en el corrent residual)

Resolució problema exemple:

Per la resolució del problema cal plantejar un balanç de matèria global:

$$m^F = m^D + m^P$$

Un balanç de matèria de sal:

$$m^F \cdot w^F sal = m^D \cdot w^D sal \cdot m^P \cdot w^P sal$$

Un balanç de matèria de la cel·la:

$$m^{F+R} = m^s + m^D$$

I un balanç de sal de la cel·la:

$$m^{F+R} \cdot w^{F+R} sal = m^S \cdot w^S sal + m^D \cdot w^D sal$$

D'aquesta forma es disposa d'un sistema de quatre equacions amb quatre incògnites apte per a la resolució.

4.3 Diagrama de interacció amb l'usuari

Es mostra a continuació en la següent figura el diagrama general de interacció amb l'usuari, on s'aprecien els diferents recorreguts que pot dur a terme l'usuari amb el programa:

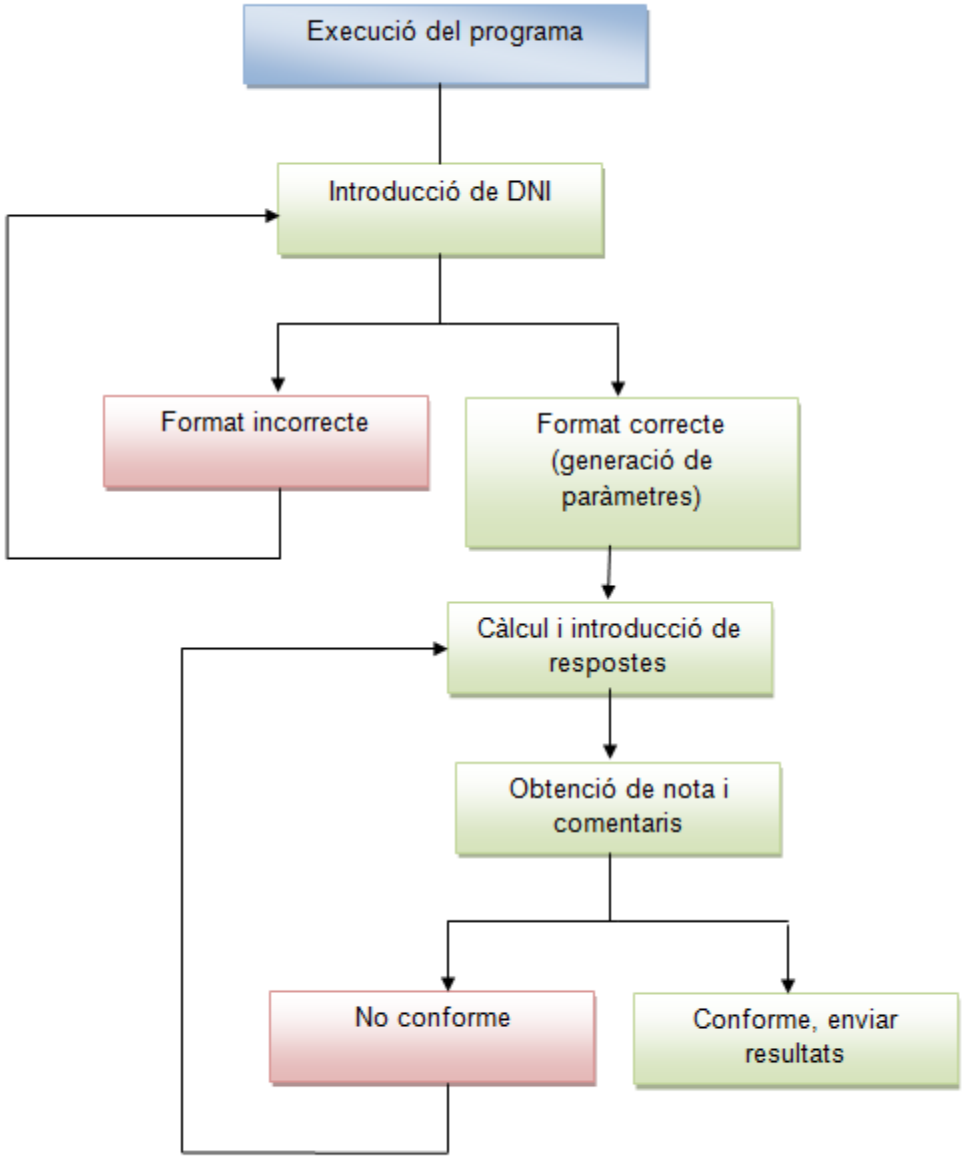

Figura 4: Diagrama de interacció amb l'usuari. Font: Pròpia.

4.4 Interfície i interacció amb l'usuari

En iniciar-se el programa, aquest té l'aspecte que es pot apreciar en la següent imatge:

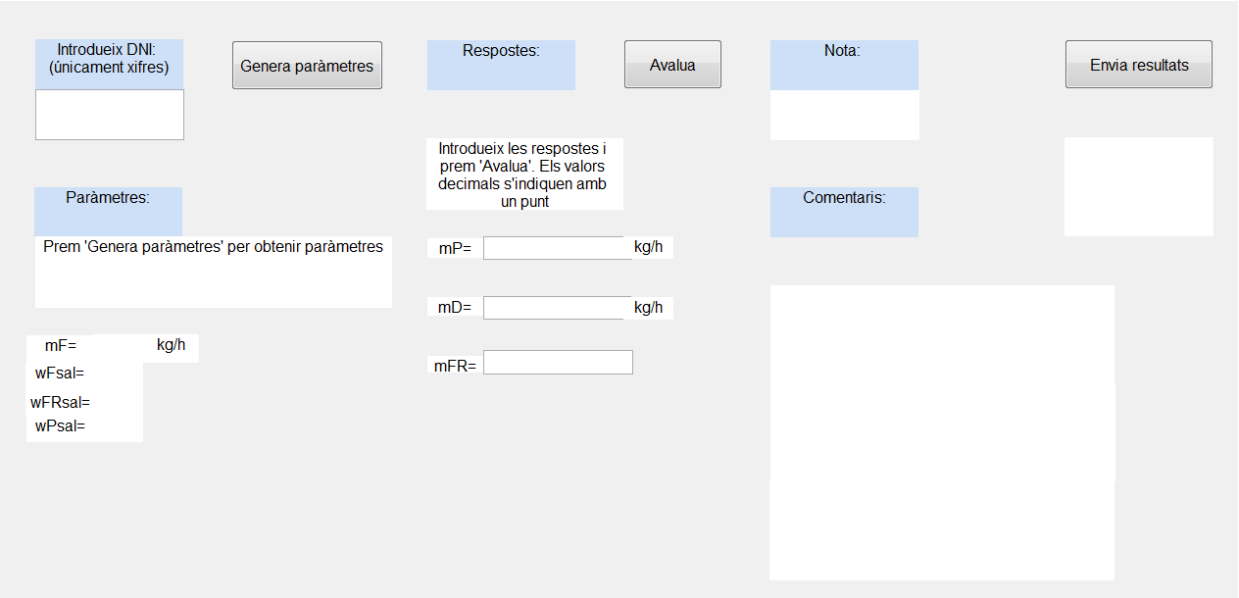

Figura 5: Aspecte inicial de la interfície. Font: Pròpia

Es pot dividir el contingut de la interfície en dos tipus d'elements segons el tipus de interacció:

- Actius: botons i caselles de text editable.
- Passius: caselles de text no editable.

Els botons que executen les funcions principals del programa són:

- Genera paràmetres
- Avalua
- Envia resultats

El primer pas per tal de disposar dels valors dels paràmetres que permetin resoldre el problema és introduir el DNI en la casella de text editable indicada i prémer el botó Genera paràmetres:

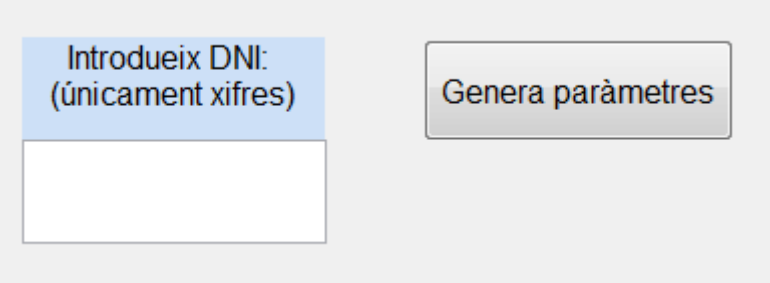

Figura 6: Introducció de DNI i botó "Genera paràmetres". Font: Pròpia

Poden mostrar-se tres missatges diferents d'error a l'hora de generar els paràmetres. El primer error possible és senzillament que no s'ha introduït cap DNI. Un altre error possible és que el format del DNI no sigui correcte. L'últim error que pot mostrar-se es dona en el cas que l'usuari ja hagi pitjat el botó "Avalua" i hagi estat avaluat amb èxit:

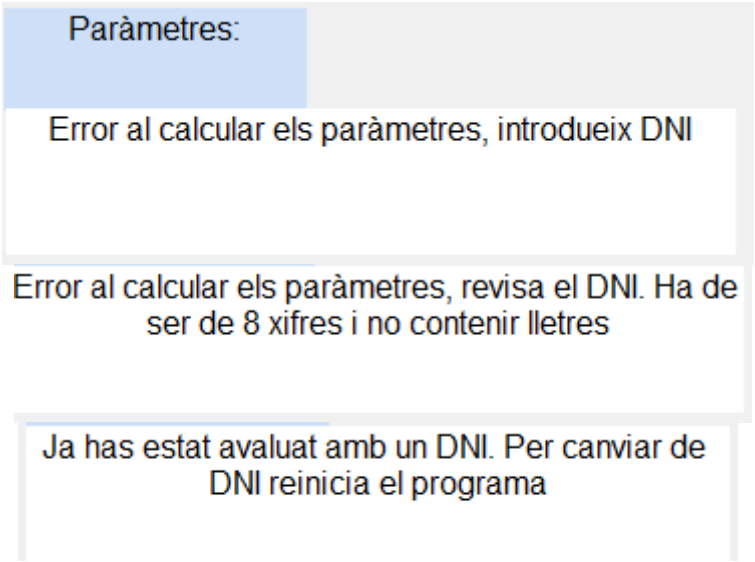

Figura 7: Errors al generar paràmetres. Font: Pròpia

En cas que el format del DNI sigui correcte es mostren els paràmetres:

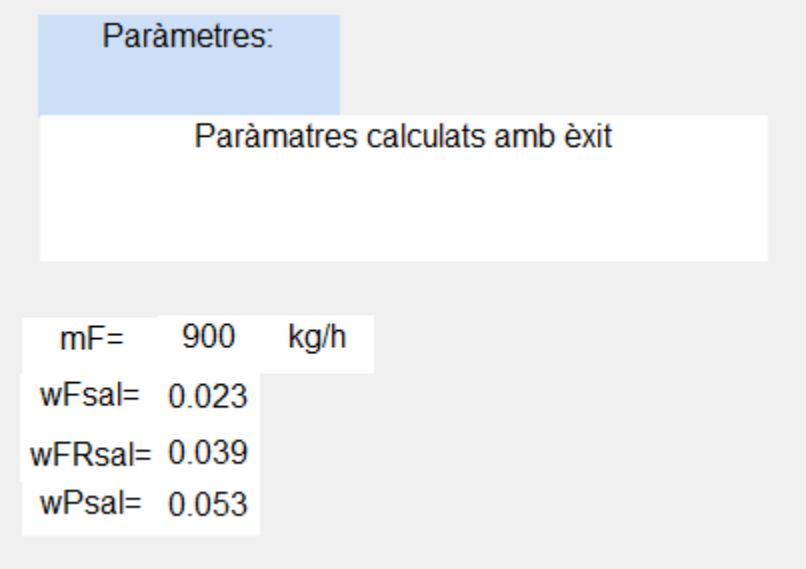

Figura 8: Paràmetres generats. Font: Pròpia

Un cop es disposa dels paràmetres l'usuari pot fer els càlculs pertinents per la resolució del problema i posteriorment introduir la resposta en les caselles corresponents:

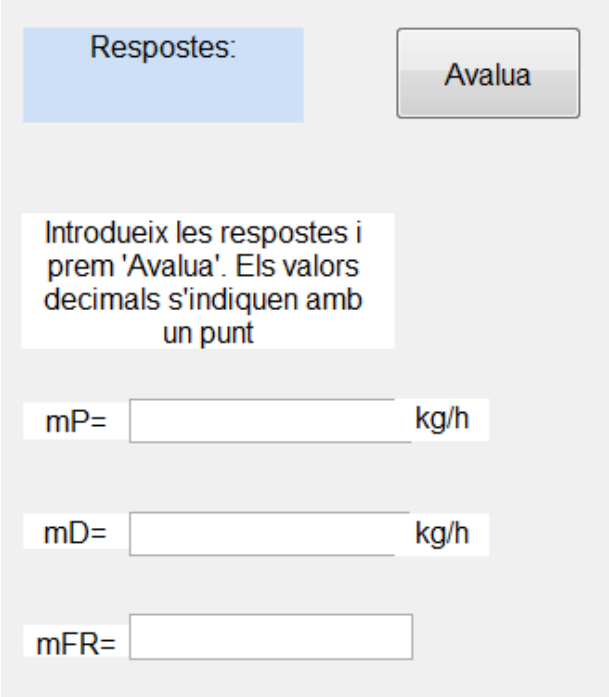

Figura 9: Introducció de respostes. Font: Pròpia

Un cop introduïdes les respostes el següent pas és pitjar el botó "Avalua". Un cop pitjat el botó es mostra la nota i els comentaris corresponents a cada qüestió del problema, havent-hi 4 possibles comentaris en funció de la resposta:

1. La resposta és correcta.
2. La resposta és incorrecta i l'error és conegut (s'explica l'error comés).
3. La resposta és incorrecta i l'error és desconegut.
4. No s'ha introduït resposta.

En l'exemple que es mostra en la següent figura, la primera resposta és incorrecta i l'error és conegut, la segona resposta és incorrecta i l'error és desconegut i en la tercera no s'ha introduït resposta:

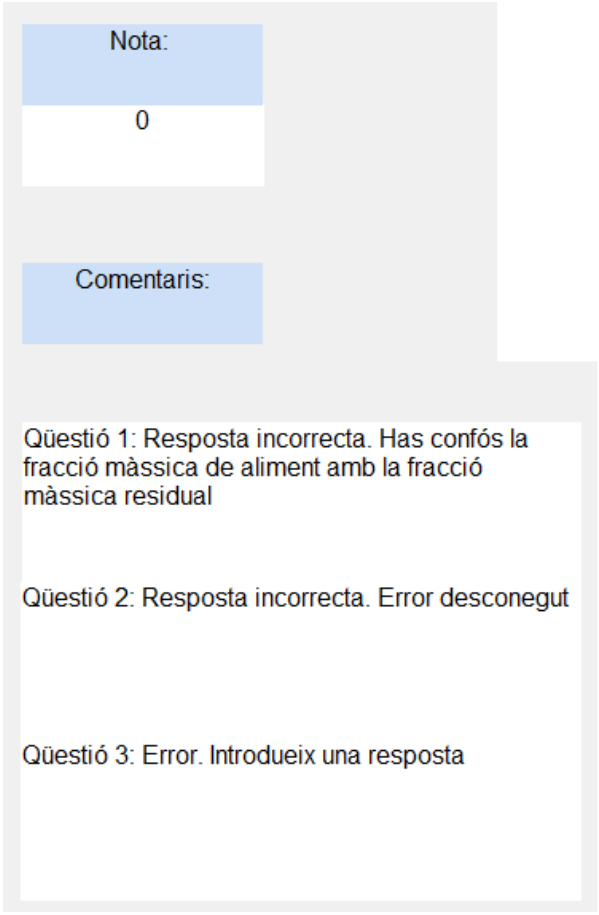

Figura 10: Nota i comentaris. Font: Pròpia

Finalment sols resta prémer el botó "Envia resultats". En la interfície es mostrarà si els resultats han pogut ser penjats o no. En cas de no poder ser penjats s'especifica si és tracta d'un error de comunicació amb la base de dades o si encara no ha estat avaluat:

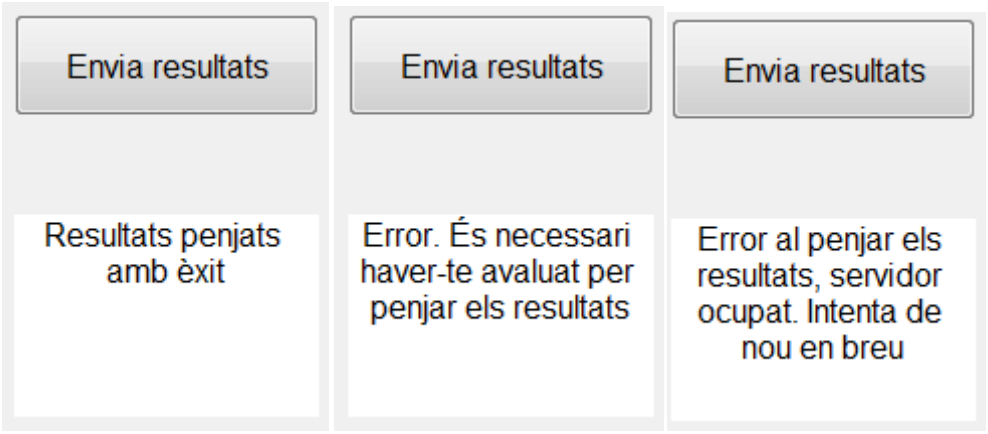

Figura 11: Nota i comentaris. Font: Pròpia

4.5 Diagrama de funcionament del codi

El funcionament del codi es defineix per quatre blocs principals: el que es corre en executar el programa i els que es corren en pitjar un dels tres botons ("Genera paràmetres", "Avalua" i "Envia resultats"). En la figura que es mostra a continuació s'exposen a trets generals les diferents operacions que es duen a terme al executar el programa i pitjar cada botó:

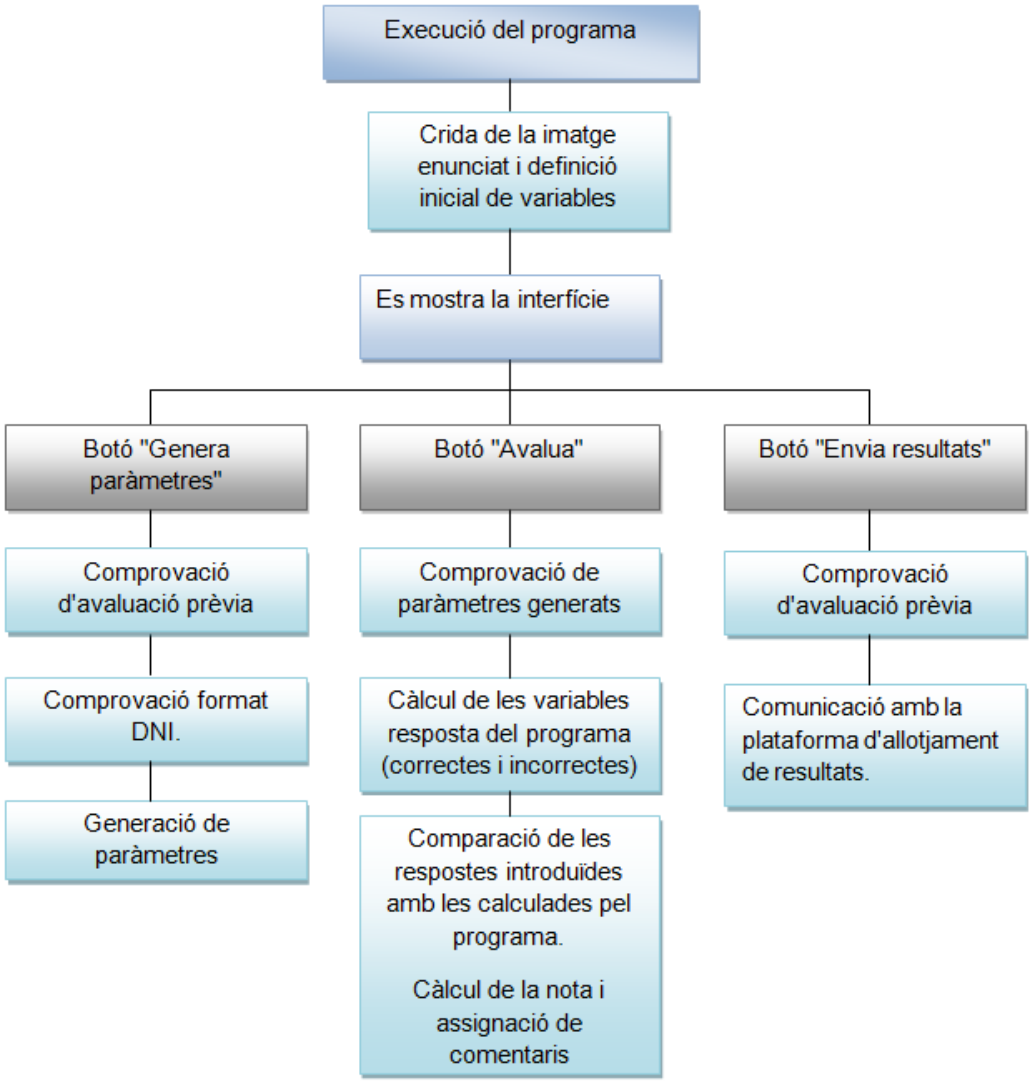

Figura 12: Diagrama de funcionament del codi. Font: Pròpia

4.6 Conceptes previs de Matlab, GUIDE i ThingSpeak

S'expliquen a continuació una sèrie de funcions i conceptes de Matlab, GUIDE i ThingSpeak per tal de comprendre correctament el codi.

4.6.1 Comunicació del codi amb la interfície

- **Funció *guidata*:**

Part de la comunicació del codi amb la interfície es dona mitjançant la funció *guidata*.

guidata permet guardar o obtenir informació de la interfície. Aquesta informació es guarda a *handles,* que es una estructura que guarda les variables i la seva associació a una funció.

El codi del programa es troba dividit en funcions independents associades al botó que les activa. És mitjançant la funció *guidata* i l'estructura *handles* que variables que han estat generades en una funció poden ser utilitzades en una altra.

Per exemple, per tal d'incorporar una variable generada en la funció associada al botó "Genera paràmetres" a l'estructura *handles* la sintaxis de la variable segueix el format:

```
handles.parametre1 = 0
```

Un cop generada la variable i inclosa en l'estructura *handles* s'utilitza la funció *guidata* per tal de guardar els canvis produïts en l'estructura *handles*. La funció *guidata* s'executa mitjançant el comandament:

```
guidata(hObject,handles)
```

Aquest comandament precisa com a input a més de l'estructura *handles* l'argument *hObject*, que és el nexe amb l'element de la interfície que ha efectuat la crida.

Un cop la variable és guardada en l'estructura es pot utilitzar en funcions associades en altres botons. Seguint l'exemple, per tal de realitzar un càlcul amb la variable "paramatre1" en la funció associada al botó "Avalua" , la sintaxis segueix el model:

```
a = handles.parametre 1 + 2

ans = 2
```

- **Funcions *get* i *set*:**

Pel que fa a la comunicació del codi amb el text que introdueix i es mostra a l'usuari s'utilitzen dues funcions: *get* i *set*.

Per obtenir l'input introduït per l'usuari en una casella de text editable de la interfície s'utilitza la funció *get,* de sintaxis:

```
get (H, Name)
```

On "H" és l'objecte d'on es vol agafar o assignar informació (per referir-se a les caselles s'utilitza el seu "tag" o etiqueta, definit a l'hora de crear la interfície) i "Name" la propietat del contingut (sempre es treballa amb tipus string).

En canvi per assignar el contingut d'una casella ja sigui de text editable o no s'utilitza la funció *set*, de sintaxis:

```
set (H,Name,Value)
```

On *Value* és el valor que s'hi vol assignar.

4.6.2 Càlculs

- **Funcions *solve* i *syms*:**

Per resoldre les equacions o sistemes d'equacions que es puguin plantejar en la resolució del problema s'utilitzen les funcions *solve* i *syms. solve* resol el sistema i *syms* defineix les variables.

Partint del següent sistema d'equacions exemple:

$$\begin{cases} 2 \cdot u + v = 0 \\ u - v = 1 \end{cases}$$

La sintaxis seria tal com:

```
syms u v
sol = solve([2*u + v == 0, u - v == 1], [u, v])
```

I un cop resolt es defineixen les variables resposta:

```
u = sol.u
v = sol.v
```

- **Funció *abs*:**

A l'hora de comparar les respostes calculades pel programa amb les respostes introduïdes per l'usuari es calcula la diferència absoluta entre el valor de les respostes. Això és fa mitjançant la funció *abs.* Per calcular la diferència absoluta entre les variables exemple A i B i assignar-la a una variable de nom difAB la sintaxis seria tal com:

```
difAB = abs (A-B)
```

4.6.3 Adequació de format

- **Funció *str2num*:**

La funció *str2num* permet canviar el tipus d'un element de cadena de caràcters a numèric. S'utilitza per poder treballar numèricament amb les dades que s'obtenen de la interfície, ja que al obtenir-les amb la funció get aquestes s'obtenen com a tipus cadena.

S'utilitza per exemple al calcular els paràmetres amb les xifres del DNI:

```
mF=str2num(DNI(4))*100;
```

En aquest cas el paràmetre mF prendrà el valor del producte de la quarta xifra del DNI per 100.

- **Funció *double*:**

La funció *double (x)* retorna un valor de doble precisió per x. S'utilitza per adequar el format dels resultats obtinguts en la resolució del programa.

- **Funció *round*:**

La funció round s'utilitza per arrodonir un número amb les xifres decimals que es desitgi:

```
Y = round(X,N)
```

Essent *X* el nombre a arrodonir, *N* el nombre de decimals i *Y* la variable de sortida.

S'utilitza per adequar el format de la nota abans de mostrar-se a la interfície.

- **Funció *table*:**

La funció *table* genera una taula a partir de les matrius introduïdes:

```
table (DNI, errorh)
```

S'utilitza en la segona avaluació de les dades per tal de generar una taula amb els DNIs i el resultat de la comparació amb l'error identificat per tal d'interpretar el resultat obtingut. La taula generada mostra un aspecte tal com:

DNI	errorh
39395908	0
39395908	0

Figura 13: Taula d'error en segona avaluació. Font: Pròpia

- **Funció *bar*:**

La funció *bar* genera un gràfic de barres per a la variable introduïda:

```
bar (hnotes)
```

S'utilitza a ThingSpeak per visualitzar les dades emmagatzemades.

4.6.4 Comprovacions

- **Funció *isempty*:**

La funció *isempty* S'utilitza per comprovar si les caselles de resposta es troben buides. Per exemple:

```
mPempty = isempty (get (handles.edit2, 'String'));
```

En la casella de "tag" edit2 s'introdueix el valor de la resposta mP. En cas que la resposta sigui inexistent la variable mPempty prendrà el valor de 1, mentre que si hi ha una resposta el seu valor serà 0.

- **Funció *size*:**

La funció *size* s'utilitza per comprovar el format del DNI. Per tal de comprovar que en el DNI introduït no s'hi troben lletres el que es fa és canviar el format de cada caràcter del DNI a numèric amb la funció *str2num*. En cas de que el caràcter fos una lletra, aquest prendrà un valor buit. D'aquesta manera al comprovar la mida del resultat de canviar el format *size* donarà 1 per cada número i 0 per cada lletra.

El resultat de la comprovació es guarda en la variable dniv, que contindrà la quantitat de números del DNI:

```
DNIv = size (str2num (DNI(1)))+size (str2num (DNI(2)))+size (str2num
(DNI(3)))+size (str2num (DNI(4)))+size (str2num (DNI(5)))+size
(str2num (DNI(6)))+size (str2num (DNI(7)))+size (str2num (DNI(8)))
```

- **Funció *numel*:**

La funció *numel* torna el nombre d'elements que conté una una matriu. La seva sintaxis és tal com:

```
n = numel(A)
```

S'utilitza a ThingSpeak per comptar el nombre d'entrades.

4.6.5 Condicionals

- **Funció *if:***

*if (*juntament amb *elseif* i *else)* és una estructura condicional que permet que es dugin a terme determinades accions quan es compleix una condició. La seva sintaxis és tal com:

```
if expression
    statements
elseif expression
    statements
else
    statements
end
```

Els blocs *elseif* i *else* son opcionals, executant-se només si les expressions anteriors són falses. Cada if pot contenir varis elseif.

4.6.6 Bucles

- **Funció *for*:**

for és una estructura de bucle que permet repetir una acció un determinat número de vegades. La seva sintaxis segueix l'exemple:

```
for i = 1 : n
    a=a+1
end
```

S'utilitza a ThingSpeak per analitzar les dades emmagatzemades.

4.6.7 Anotacions

Per tal de fer anotacions al llarg del codi, s'utilitza el símbol % per indicar que és una línia de codi no executable. Les anotacions tenen un aspecte tal com:

```
% Aquest és un comentari de mostra.
```

4.6.8 ThingSpeak

- **Funció *webwrite* i *response*:**

Per enviar els resultats a la base de dades s'utilitza la funció *webwrite* mentre que la resposta que s'obté del seu ús es guarda en la variable *response*. La seva sintaxis es tal com:

```
response =
webwrite(thingSpeakWriteURL,'api_key',writeApiKey,fieldName1,fieldValu
e1,fieldName2,fieldValue2,fieldName3,fieldValue3,fieldName4,fieldValue
4,fieldName5,fieldValue5,fieldName6,fieldValue6)
```

Els inputs de la funció *webwrite* són la URL de ThingSpeak, les claus de permís d'escriptura, el nom dels camps a omplir i el seu valor.

En cas que la comunicació sigui correcta, *response* guardarà el valor corresponent al nombre de l'entrada, és a dir, si ja hi ha 43 files de dades guardades r*esponse* prendrà el valor de 44, mentre que si la comunicació ha estat errònia prendrà el valor 0.

- **Funció *thingSpeakRead*:**

La funció *thingSpeakRead* s'utilitza per obtenir informació emmagatzemada a ThingSpeak. La seva sintaxis és tal com:

```
A = thingSpeakRead(readChannelID1, 'Field', 1, 'NumPoints', 30, 'ReadKey', readAPIKey1);
```

La informació a ThingSpeak es guarda en canals i cada canal es divideix en camps, de manera que per tal de llegir la informació cal especificar en els inputs de la funció la ID del canal que es vol llegir, el número del camp, la quantitat d'entrades que es volen llegir i la clau de lectura. La informació llegida quedarà guardada en la variable A.

4.7 Taula de variables

A continuació es mostra una taula amb totes les variables que participen en el programa exemple amb una breu descripció de cada una i el valor que prenen en les diferents funcions al llarg del recorregut del problema exemple.

Per simplificar i facilitar la comprensió s'omet en el nom de la variable la sintaxis de l'estructura *handles* (ex: en comptes de handles.DNI, es mostra simplement DNI):

Taula 1: Variables "Funció inicial" i "Genera paràmetres". Font: Pròpia

Funció	Nom de la variable	Descripció del contingut	Valor Problema Exemple
Funció Inicial	parametres	Definició inicial, variables de comprovació	0
	avaluat		0
Genera Paràmatres	DNI	DNI introduït per l'usuari	39395908
	DNIv	Nombre de xifres del DNI introduït	8
	mF	Paràmatres generats a partir de xifres del DNI	DNI(4)·100 = 900
	wFsal		0,02+DNI(1)/1000 = 0,023
	wFRsal		0,03+DNI(2)/1000 = 0,039
	wPsal		0,05+DNI(3)/1000 = 0,053
	parametres	Comprovació de generació de paràmatres	1

Taula 2:Variables "Avalua" 1.0 . Font: Pròpia

Funció	Nom de la variable	Descripció del contingut	Valor Problema Exemple
	nota	Definició inicial de les variables d'errors	0
	ea		0
	eb		0
	ec		0
	ed		0
	ee		0
	ef		0
	eg		0
	mP	Respostes introduïdes per l'usuari	2074
	mD		800
	mFR		-
	eq1	Equacions per la resolució del problema	mF = pmD + pmP
	eq2		mF·wFsal = pmP·wPsal
	eq3		pmFR = pmS + pmD
	eq4		pmFR·wFRsal = pmS·wPsal
	sol	Conté la resolució del sistema d'equacions	pmP, pmD, pmFR, pmS
	pmP	Respostes correctes calculades pel programa	390,57
	pmD		509,43
	pmFR		1928,57
	pmS		1419,14
	pmP1	Resposta incorrecta calculada pel programa	2074

Taula 3: Variables "Avalua" 2.0 . Font: Pròpia

Funció	Nom de la variable	Descripció del contingut	Valor Problema Exemple
Avalua	difmP	Diferència absoluta entre la resposta calculada correcta i la introduïda	pmP-mP = 1683,43
	difmD		pmD -mD = 290,57
	difFR		pmFR-mFR = 1928,57
	difmP1	Diferència absoluta entre la resposta calculada incorrecta i la introduïda	pmP1-mP = 0
	mPempty	Comprovacions per si s'ha introduït respostes o no	0
	mDempty		0
	mFRempty		1
	epsilon	Tolerància	0,05
	C1	Comentaris de les diferents qüestions i errors associats	Qüestió 1: Resposta incorrecta. Has confós la fracció màssica de aliment amb la fracció màssica residual.
	eg		1
	C2		Qüestió 2: Resposta incorrecta. Error desconegut
	ee		1
	C3		Qüestió 3: Error. Introdueix una resposta
	ec		1
	nota	Nota de l'avaluació	0
	avaluat	Confirmació d'avaluació	1

Taula 4: Variables "Envia resultats" 1.0. Font: Pròpia

Funció	Nom de la variable	Descripció del contingut	Valor Problema Exemple
Envia resultats	thingSpeakURL	URL de ThingSpeak	http://api.thingspeak.com/
	thingSpeakWriteURL	URL d'escriptura de Thingspeak	[thingSpeakURL 'update']
	writeApiKey1	Clau API d'escriptura del canal 1	QPU3TPGSJOPOOJ5W
	fieldName11	Nom del primer camp (canal 1)	DNI
	fieldValue11	Contingut del primer camp (canal 1)	DNI ('39395908')
	fieldName21	Nom del segon camp (canal 1)	nota
	fieldValue21	Contingut del segon camp (canal 1)	nota (0)
	fieldName31	Nom del tercer camp (canal 1)	mP
	fieldValue31	Contingut del tercer camp	mP (2074)
	fieldName41	Nom del quart camp (canal 1)	mD
	fieldValue41	Contingut del quart camp (canal 1)	mD (800)
	fieldName51	Nom del cinquè camp (canal 1)	mFR
	fieldValue51	Contingut del cinquè camp (canal 1)	mFR (0)
	writeApiKey2	Clau API d'escriptura del canal 2	XZRRM71M72OY8FUZ
	fieldName12	Nom del primer camp (canal 2)	ea
	fieldValue12	Contingut del primer camp (canal 2)	0
	fieldName22	Nom del segon camp (canal 2)	eb
	fieldValue22	Contingut del segon camp (canal 2)	0
	fieldName32	Nom del tercer camp (canal 2)	ec
	fieldValue32	Contingut del tercer camp (canal 2)	1
	fieldName42	Nom del quart camp (canal 2)	ed
	fieldValue42	Contingut del quart camp (canal 2)	0
	fieldName52	Nom del cinquè camp (canal 2)	ee
	fieldValue52	Contingut del cinquè camp (canal 2)	1
	fieldName62	Nom del sisè camp (canal 2)	ef
	fieldValue62	Contingut del sisè camp (canal 2)	0
	fieldName72	Nom del setè camp (canal 2)	eg
	fieldValue72	Contingut del setè camp (canal 2)	1

Taula 5: Variables "Envia resultats" 2.0. Font: Pròpia

Funció	Nom de la variable	Descripció del contingut	Valor Problema Exemple
Envia resultats	writeApiKey3	Clau API d'escriptura del canal 3	XZRRM71M72OY8FUZ
	fieldName13	Nom del primer camp (canal 3)	ea
	fieldValue13	Contingut del primer camp (canal 3)	0
	fieldName23	Nom del segon camp (canal 3)	eb
	fieldValue23	Contingut del segon camp (canal 3)	0
	fieldName33	Nom del tercer camp (canal 3)	ec
	fieldValue33	Contingut del tercer camp (canal 3)	1
	fieldName43	Nom del quart camp (canal 3)	ed
	fieldValue43	Contingut del quart camp (canal 3)	0
	response1	Resposta del canal 1	1
	response2	Resposta del canal 2	1
	response3	Resposta del canal 3	1

4.8 Identificació del codi corresponent a cada funció i codi en detall

4.8.1 Funcions que no participen en la resolució del problema

La primera part del codi consisteix en fragments de codi generats de forma automàtica al crear la interfície amb GUIDE, prescindibles per la resolució del problema però necessaris pel correcte funcionament del programa:

```
function varargout = ProblemaExemple(varargin)


gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ProblemaExemple_OpeningFcn, ...
                   'gui_OutputFcn',  @ProblemaExemple_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end


if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

La següent funció serveix per obtenir sortides del programa a la línia de comandaments de Matlab. Per com està dissenyat el programa és una funció que no resulta útil:

```
function varargout = ProblemaExemple_OutputFcn(hObject, eventdata,
handles)
varargout{1} = handles.output;
```

Les següents funcions es corresponen a les diferents caselles de text editable del programa. A cada casella li corresponen dues funcions. La funció "Callback", que executa el codi que conté la casella quan aquesta es cridada i la funció "CreateFcn", que executa el codi quan l'element és creat al executar el programa però abans de ser mostrat.

De nou es tracta de funcions que no participen en la resolució, ja que tot el codi es troba en les funcions de crida dels diferents botons.

```
function cDNI_Callback(hObject, eventdata, handles)
```

```
function cDNI_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function cmP_Callback(hObject, eventdata, handles)
```

```
function cmP_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function cmD_Callback(hObject, eventdata, handles)
```

```
function cmD_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function cmFR_Callback(hObject, eventdata, handles)
```

```
function cmFR_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

4.8.2 Funció "Funció obertura"

La següent funció s'executa abans de que la interfície es faci visible. S'utilitza per definir les variables "parametres" i "avaluat":

```
function ProblemaExemple_OpeningFcn(hObject, eventdata, handles,
varargin)
```

- La primera línia de la funció es genera de forma automàtica al dissenyar la interfície. Compleix funcions de comunicació entre funcions del programa

```
handles.output = hObject;
```

- Es defineixen les variables parametres i avaluat:

```
handles.parametres=0;
handles.avaluat=0;
```

- I per acabar s'actualitza l'estructura handles:

```
guidata(hObject, handles);
```

4.8.3 Funció "Genera paràmetres"

La següent funció s'executa al prémer el botó "Genera paràmetres".

El primer que fa es comprovar si l'usuari ja ha estat avaluat. En cas que no ho hagi estat es mostra un missatge d'error i no es generen paràmetres. En cas que l'usuari no hagi estat avaluat encara es guarda el DNI introduït per l'usuari.

Seguidament comprova que el DNI introduït compleix els requeriments de format. En cas de no ser-ho mostra un missatge d'alerta a l'usuari i no es genera cap paràmetre. En cas que el format sigui correcte es generen els paràmetres.

```
function Genera_Parametres_Callback(hObject, eventdata, handles)
```

- Es comprova si l'usuari ha estat avaluat. En cas afirmatiu es mostra el missatge d'error corresponent:

```
if handles.avaluat == 1

    set (handles.cParametres, 'String', 'Ja has estat avaluat amb un
DNI. Per canviar de DNI reinicia el programa');
```

- En cas que no hagi estat avaluat encara es guarda el DNI:

```
else

    handles.DNI = get (handles.cDNI, 'String');
```

- En cas que no s'hagi introduït cap DNI es mostra el missatge d'error pertinent i s'assigna un espai buit a les caselles de paràmetres:

```
    if length(handles.DNI) == 0

        set (handles.cParametres, 'String', 'Error al calcular els
paràmetres, introdueix DNI')
        set (handles.cmF, 'String', '');
        set (handles.cwFsal, 'String', '');
        set (handles.cwFRsal, 'String', '');
        set (handles.cwPsal, 'String', '');
```

- Si el nombre de caràcters del DNI és l'adequat es procedeix a comprovar si tots els caràcters són xifres:

```
elseif length(handles.DNI) == 8

        DNIv = size (str2num (handles.DNI(1)))+size (str2num
(handles.DNI(2)))+size (str2num (handles.DNI(3)))+size (str2num
(handles.DNI(4)))+size (str2num (handles.DNI(5)))+size (str2num
(handles.DNI(6)))+size (str2num (handles.DNI(7)))+size (str2num
(handles.DNI(8)));
```

- En cas que el format sigui l'adequat es calculen els paràmetres i s'assignen a la casella corresponent. S'assigna també el missatge d'èxit corresponent a la interfície i s'actualitza la variable "parametres" conforme aquests han estat generats.

```
if DNIv == 8

    handles.mF=str2num(handles.DNI(4))*100;
    handles.wFsal=0.02+str2num(handles.DNI(1))/1000;
    handles.wFRsal=0.03+str2num(handles.DNI(2))/1000;
    handles.wPsal=0.05+str2num(handles.DNI(3))/1000;

    set (handles.cmF, 'String', handles.mF);
    set (handles.cwFsal, 'String', handles.wFsal);
    set (handles.cwFRsal, 'String', handles.wFRsal);
    set (handles.cwPsal, 'String', handles.wPsal);

    set (handles.cParametres, 'String', 'Paràmatres calculats
amb èxit')

    handles.parametres = 1;
```

- En cas que els 8 caràcters no siguin números s'actualitza la interfície amb el missatge d'error corresponent:

```
else
    set (handles.cParametres, 'String', 'Error al calcular els
paràmetres, revisa el DNI. Ha de ser de 8 xifres i no contenir
lletres')
    set (handles.cmF, 'String', '');
    set (handles.cwFsal, 'String', '');
    set (handles.cwFRsal, 'String', '');
    set (handles.cwPsal, 'String', '');
end
```

- En cas que el DNI introduït no contingui 8 caràcters es procedeix amb el missatge d'error:

```
    else

        set (handles.cParametres, 'String', 'Error al calcular els
paràmetres, revisa el DNI. Ha de ser de 8 xifres i no contenir
lletres')
        set (handles.cmF, 'String', '');
        set (handles.cwFsal, 'String', '');
        set (handles.cwFRsal, 'String', '');
        set (handles.cwPsal, 'String', '');

    end
```

- S'actualitza l'estructura i acaba la funció:

```
    guidata(hObject, handles);

end
```

4.8.4 Funció "Avalua "

La següent funció s'executa al prémer el botó "Avalua".

És la funció encarregada de calcular els resultats correctes del problema i comparar-los amb les respostes introduïdes, tot analitzant els possibles errors comesos. També calcula la nota i la mostra juntament amb els comentaris sobre la resolució.

```
function Avalua_Callback(hObject, eventdata, handles)
```

- Primer de tot es defineixen les variables dels errors i de la nota, assegurant d'aquesta manera que el valor que contenen al final de la funció es correspon a l'avaluació realitzada:

```
handles.nota=0;
handles.ea = 0;
handles.eb = 0;
handles.ec = 0;
handles.ed = 0;
handles.ee = 0;
handles.ef = 0;
handles.eg = 0;
```

- Seguidament es comprova que els paràmetres han estat generats:

```
if handles.parametres == 1
```

- En cas afirmatiu s'agafen les respostes introduïdes per l'usuari i es dona la resolució numèrica del problema.

  En aquest cas exemple la resolució es tracta d'un sistema d'equacions lineals i per tal de resoldre'l es defineixen primer les variables que es volen calcular com a variables simbòliques i després les equacions.

  Posteriorment és resol el sistema mitjançant la funció "solve" i es guarden les respostes a la variable "sol":

```
handles.mP = str2num(get (handles.cmP, 'String'));
handles.mD = str2num(get (handles.cmD, 'String'));
handles.mFR = str2num(get (handles.cmFR, 'String'));

syms pmP pmD pmFR pmS

eq1 = handles.mF == pmD + pmP;
eq2 = handles.mF*handles.wFsal == pmP*handles.wPsal;
eq3 = pmFR == pmS + pmD;
eq4 = pmFR*handles.wFRsal == pmS*handles.wPsal;

sol = solve([eq1, eq2, eq3, eq4], [pmP, pmD, pmFR, pmS]);
```

- Un cop resolt s'assigna el valor de les respostes guardat en la variable "sol" a les variables resposta individuals:

```
handles.pmP = double(sol.pmP);
handles.pmD = double(sol.pmD);
handles.pmFR = double(sol.pmFR);
handles.pmS = double(sol.pmS);
```

- Seguidament es calculen les respostes errònies dels errors freqüents coneguts. En aquest problema exemple l'error freqüent conegut és el de confondre la fracció màssica del corrent d'aliment amb la fracció màssica del corrent residual:

```
pmP1 = (handles.mF*handles.wPsal)/handles.wFsal;
```

- Es calcula la diferència absoluta entre la resposta correcta i la resposta introduïda:

```
difmP = abs(pmP-handles.mP);
difmD = abs(pmD-handles.mD);
difFR = abs(pmFR-handles.mFR);
```

- I també la diferència absoluta entre els errors freqüents i les respostes introduïdes:

```
difmP1 = abs(pmP1-handles.mP);
```

- Posteriorment es comprova si les caselles de resposta estan buides:

```
mPempty = isempty (get (handles.cmP, 'String'));
mDempty = isempty (get (handles.cmD, 'String'));
mFRempty = isempty (get (handles.cmFR, 'String'));
```

- Es defineix també la tolerància permesa, en aquest cas d'un 5%:

```
epsilon = 0.05;
```

- Un cop es disposa de les respostes correctes, errors coneguts, respostes introduïdes, les seves diferències absolutes i la tolerància comença el recorregut condicional per comparar les respostes.

  Es considera que la resposta es correcta quan la diferència absoluta entre la resposta correcta i la introduïda és inferior al producte de la tolerància i la resposta correcta.

  Si la condició es compleix s'assigna el comentari corresponent i es suma el valor de la qüestió a la nota:

```
if difmP<epsilon*pmP

    C1  = 'Qüestió 1: Resposta correcta';

    handles.nota = handles.nota+1;
```

- En cas que la resposta no sigui correcta es comprova si s'ha comès algun dels errors freqüents coneguts. Per realitzar la comprovació s'utilitza el mateix criteri que quan es compara amb la resposta correcta.

  En cas que s'hagi comés l'error el comentari explica l'error comés. També en cas afirmatiu es canvia el valor de la variable de l'error corresponent a 1:

```
elseif difmP1< epsilon*pmP1

    C1 = 'Qüestió 1: Resposta incorrecta. Has confós la fracció
màssica de aliment amb la fracció màssica residual';

    handles.eg = 1;
```

- Es comprova també si la resposta és buida i en cas afirmatiu s'assigna el comentari i l'error corresponent:

```
elseif mPempty == 1

    C1= 'Qüestió 1: Error. Introdueix una resposta';

    handles.ea = 1;
```

- En cas que la resposta no coincideixi amb cap dels casos comparats s'indica en el comentari que l'error és desconegut i s'assigna l'error corresponent:

```
else

    C1 = 'Qüestió 1: Resposta incorrecta. Error desconegut';

     handles.ed = 1;

end
```

- El procediment és el mateix per les altres qüestions. Per la qüestió 2:

```
if difmD<epsilon*pmD

    C2 ='Qüestió 2: Resposta correcta';

    handles.nota = handles.nota+1;

elseif mDempty == 1

    C2 = 'Qüestió 2: Error. Introdueix una resposta';

    handles.eb = 1;

else

    C2 ='Qüestió 2: Resposta incorrecta. Error desconegut';

    handles.ee = 1;

end
```

– I per la qüestió 3:

```
if difFR<epsilon*pmFR
    C3  ='Qüestió 3: Resposta correcta';
     handles.nota = handles.nota+1;

elseif mFRempty == 1
    C3 = 'Qüestió 3: Error. Introdueix una resposta';
    handles.ec = 1;

else
    C3 ='Qüestió 3: Resposta incorrecta. Error desconegut';
    handles.ef = 1;


end
```

– Seguidament es pondera la nota i s'arrodoneix:

```
    handles.nota = 10*handles.nota/3;
    handles.nota = round(handles.nota, 2);
```

– S'actualitza la variable avaluat conforme s'ha avaluat, s'actualitza l'estructura i s'assigna la nota i els comentaris a les caselles corresponents:

```
    handles.avaluat = 1;


    guidata(hObject, handles);


    set (handles.cNota, 'String', handles.nota)
    set (handles.cC1, 'String', C1)
    set (handles.cC2, 'String', C2)
    set (handles.cC3, 'String', C3)
    set (handles.cRespostes, 'String', 'Avaluació amb èxit')
```

- Per acabar es posa fi al condicional de comprovació de paràmetres generats:

```
else

    set (handles.cRespostes, 'String', 'No es disposa de paràmatres
per a poder avaluar')

end
```

4.8.5 Funció "Envia resultats"

En l'última funció és on es dona la comunicació amb la base de dades i s'executa al prémer el botó "Envia resultats":

```
function EnviaResultats_Callback(hObject, eventdata, handles)
```

- Es comença comprovant si l'usuari ha estat avaluat:

```
if handles.avaluat == 1
```

- En cas afirmatiu, es defineix la URL de la pàgina de la base de dades:

```
thingSpeakURL = 'http://api.thingspeak.com/';
thingSpeakWriteURL = [thingSpeakURL 'update'];
```

- Es defineixen la clau d'escriptura, els diferents camps i el seu valor per al canal 1, on es guarden el DNI, la nota i els valors de les respostes introduïdes per l'usuari:

```
writeApiKey1 = 'QPU3TPGSJOPOOJ5W';

fieldName11 = 'field1';
fieldValue11 = handles.DNI;

fieldName21 = 'field2';
fieldValue21 = handles.nota;

fieldName31 = 'field3';
fieldValue31 = handles.mP;

fieldName41 = 'field4';
fieldValue41 = handles.mD;

fieldName51 = 'field5';
fieldValue51 = handles.mFR;
```

– Es defineixen la clau d'escriptura, els diferents camps i el seu valor pel canal 2, on es guarden els errors:

```
writeApiKey2 = 'XZRRM71M72OY8FUZ';

fieldName12 = 'field1';
fieldValue12 = handles.ea;

fieldName22 = 'field2';
fieldValue22 = handles.eb;

fieldName32 = 'field3';
fieldValue32 = handles.ec;

fieldName42 = 'field4';
fieldValue42 = handles.ed;

fieldName52 = 'field5';
fieldValue52 = handles.ee;

fieldName62 = 'field6';
fieldValue62 = handles.ef;

fieldName72 = 'field7';
fieldValue72 = handles.eg;
```

– I es defineixen també la clau d'escriptura, els diferents camps i el seu valor pel canal 3, on es guarden les respostes calculades pel programa:

```
writeApiKey3 = '6B6CDIRFL40Z0B2Z';

fieldName13 = 'field1';
fieldValue13 = handles.pmP;

fieldName23 = 'field2';
fieldValue23 = handles.pmD;

fieldName33 = 'field3';
fieldValue33 = handles.pmFR;

fieldName43 = 'field4';
fieldValue43 = handles.pmS;
```

– S'aplica la funció *webrite* pels tres canals:

```
    response1 =
webwrite(thingSpeakWriteURL,'api_key',writeApiKey1,fieldName11,fieldVa
lue11,fieldName21,fieldValue21,fieldName31,fieldValue31,fieldName41,fi
eldValue41,fieldName51,fieldValue51);

    response2 =
webwrite(thingSpeakWriteURL,'api_key',writeApiKey2,fieldName12,fieldVa
lue12,fieldName22,fieldValue22,fieldName31,fieldValue32,fieldName42,fi
eldValue42,fieldName52,fieldValue52,fieldName62,fieldValue62,fieldName
72,fieldValue72);

response3 =
webwrite(thingSpeakWriteURL,'api_key',writeApiKey3,fieldName13,fieldVa
lue13,fieldName23,fieldValue23,fieldName33,fieldValue33,fieldName43,fi
eldValue43);
```

- S'inicia un nou condicional per comprovar si la comunicació amb la base de dades ha estat possible o no i es comunica el cas a l'usuari:

```
    if response1 == '0'

            set (handles.cEnvia, 'String', 'Error al penjar els
resultats, servidor ocupat. Intenta de nou en breu')

    elseif response2 == '0'

            set (handles.cEnvia, 'String', 'Error al penjar els
resultats, servidor ocupat. Intenta de nou en breu')

    elseif response3 == '0'

            set (handles.cEnvia, 'String', 'Error al penjar els
resultats, servidor ocupat. Intenta de nou en breu')



    else
            set (handles.cEnvia, 'String', 'Resultats penjats amb
èxit')

    end
```

- Finalment es tanca el condicional de comprovació d'avaluació prèvia:

```
else

    set (handles.cEnvia, 'String', 'Errors. És necessari haver-te
avaluat per penjar els resultats')

end
```

4.9 ThingSpeak. Emmagatzematge de resultats i tractament de dades

En aquest apartat s'explica com s'utilitza ThingSpeak en el cas del problema exemple. En el següent apartat d'aquest projecte en el qual s'aborda el disseny d'un segon problema exemple s'explica amb més profunditat com utilitzar ThingSpeak.

4.9.1 Canals

L'element clau de ThingSpeak són els canals. Aquests poden ser públics o privats i s'utilitzen per emmagatzemar les dades. Cada canal consta de:

- 8 camps per guardar dades de qualsevol tipus.
- 3 camps de localització. Pensats per guardar dades de latitud, longitud i alçada. Són camps que resulten pràctics per aplicacions que integrin serveis de localització.

En el cas del problema exemple en que s'utilitzen 3 canals per guardar les dades, es mostren a continuació els camps del canal 1 on es guarden el DNI, la nota i els resultats de l'usuari:

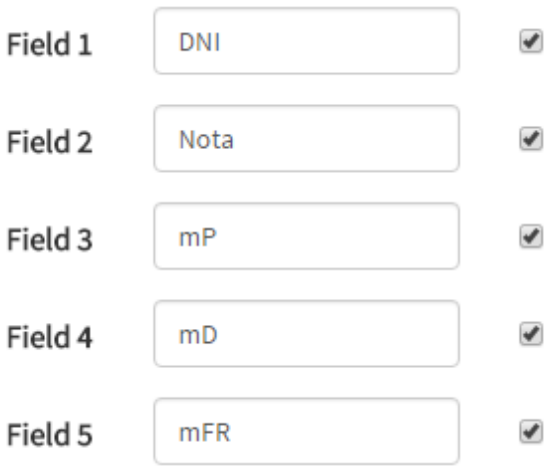

Figura 14: Canal 1 problema exemple. Font: Pròpia.

Els camps del canal 2, on es guarden els errors comesos:

Figura 15: Canal 2 problema exemple. Font: Pròpia.

I els camps del canal 3, on es guarden les respostes calculades pel programa:

Figura 16: Canal 3 problema exemple. Font: Pròpia.

Cada canal conta amb una ID identificativa i claus d'escriptura i lectura. En el cas del canal 1 la seva ID i claus són:

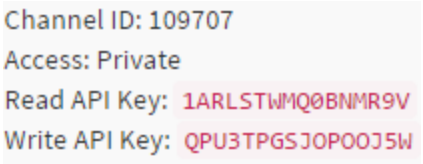

Figura 17: ID I claus canal 1 problema exemple. Font: Pròpia.

4.9.2 Exportació de dades

ThingSpeak permet exportar les dades en format JSON, XML i CSV. Al exportar-les en format CSV poden obrir-se utilitzant l'Excel i després d'adequar-ne el format es disposa de les dades del canal en format taula, on cada fila és una entrada i cada columna correspon a un camp del canal.

D'un total de 14 entrades simulades a mode d'exemple es mostren a continuació els fulls Excel de cada canal un cop adequat el seu format.

Pel canal 1 on s'hi troben guardades les dades de DNIs, notes i resultats aquesta mostra un aspecte tal com:

Taula 6: Dades exportades canal 1 problema exemple. Font: Pròpia.

created_at	entry_id	field1	field2	field3	field4	field5
2016-09-24 19:02:58 UTC	1	39395908	0			
2016-09-24 19:03:27 UTC	2	39395908	10	390.566	509.434	19.285.714
2016-09-24 19:03:43 UTC	3	39395908	10	390.566	509.434	19.285.714
2016-09-24 19:12:20 UTC	4	39395908	3.33			19.285.714
2016-09-24 19:13:28 UTC	5	39395908	3.33	0	509.434	
2016-09-24 19:14:48 UTC	6	39395908	3.33	390.566	0	0
2016-09-24 19:15:14 UTC	7	39395908	6.67		509.434	19.285.714
2016-09-24 19:15:33 UTC	8	39395908	6.67	390.566		19.285.714
2016-09-24 19:15:51 UTC	9	39395908	6.67	390.566	509.434	
2016-09-24 19:16:21 UTC	10	39395908	6.67	0	509.434	19.285.714
2016-09-24 19:17:26 UTC	11	39395908	6.67	390.566	0	19.285.714
2016-09-24 19:17:42 UTC	12	39395908	6.67	390.566	509.434	0
2016-09-24 19:18:02 UTC	13	39395908	6.67	2073.9	509.434	19.285.714
2016-09-24 19:18:24 UTC	14	39395908	6.67	509.434	509.434	19.285.714

Mentre que pel canal 2 que conté els errors la taula obtinguda mostra un aspecte tal com:

Taula 7: Dades exportades canal 2 problema exemple. Font: Pròpia.

created_at	entry_id	field1	field2	field3	field4	field5	field6	field7
2016-09-24 19:02:59 UTC	1	1	1	1	0	0	0	0
2016-09-24 19:03:28 UTC	2	0	0	0	0	0	0	0
2016-09-24 19:03:43 UTC	3	0	0	0	0	0	0	0
2016-09-24 19:12:20 UTC	4	1	1	0	0	0	0	0
2016-09-24 19:13:28 UTC	5	0	0	1	1	0	0	0
2016-09-24 19:14:48 UTC	6	0	0	0	0	1	1	0
2016-09-24 19:15:14 UTC	7	1	0	0	0	0	0	0
2016-09-24 19:15:33 UTC	8	0	1	0	0	0	0	0
2016-09-24 19:15:51 UTC	9	0	0	1	0	0	0	0
2016-09-24 19:16:21 UTC	10	0	0	0	1	0	0	0
2016-09-24 19:17:26 UTC	11	0	0	0	0	1	0	0
2016-09-24 19:17:42 UTC	12	0	0	0	0	0	1	0
2016-09-24 19:18:03 UTC	13	0	0	0	0	0	0	1
2016-09-24 19:18:25 UTC	14	0	0	0	1	0	0	0

4.9.3 Tractament de dades

Com es comenta anteriorment, a més de la possibilitat de tabular les dades ThingSpeak ofereix varis mètodes de tractament de dades mitjançant anàlisis, visualitzacions i generació de plugins que permeten mitjançant llenguatge de Matlab interpretar fàcilment com són els resultats generals del problema:

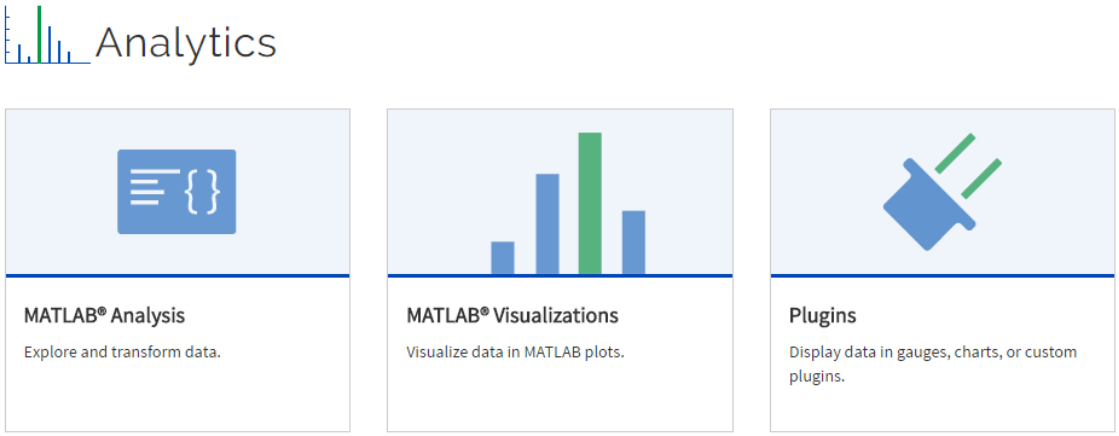

Figura 18: Opcions de tractament de dades. Font: https://thingspeak.com/apps

En el cas del problema exemple es generen dos visualitzacions en forma d'histogrames que faciliten el seguiment de les dades que entren en els canals a temps real.

El primer histograma mostra la freqüència en que es repeteix cada nota.

S'explica a continuació el codi mitjançant el qual és generat:

- Es comença definint la ID del canal que es vol llegir i la clau de lectura. La ID del canal correspon a la del canal 1:

```
readChannelID = 109707;

readAPIKey = '1ARLSTWMQ0BNMR9V';
```

- Seguidament s'utilitza la funció thingSpeakRead per guardar les notes a la variable notes:

```
notes = thingSpeakRead(readChannelID, 'Field', 2, 'NumPoints', 30, 'ReadKey', readAPIKey);
```

- Havent-hi quatre notes possibles, es defineixen quatre variables on es guardarà el compteig de les notes. A cada nota li correspondrà un número en l'eix horitzontal de l'histograma tal i com es mostra en la següent taula:

Taula 8: Correspondència entre el número d'identificació en l'eix horitzontal de l'histograma i la nota. Font: Pròpia.

Nota	Nom de la variable	Número d'identificació en l'eix horitzontal de l'histograma
0	nota1	1
3,33	nota2	2
6,67	nota3	3
10	nota4	4

```
nota1=0;

nota2=0;

nota3=0;

nota4=0;
```

- Cal definir també el nombre de iteracions necessàries pel compteig de les notes. Això es fa utilitzant la funció *numel* en la variable que conté les notes i guardant-ne el resultat en la variable n:

```
n=numel (notes);
```

- En el següent pas es recorren totes les notes guardades i s'efectua el compteig de cada nota mitjançant iteracions amb un bucle *for* :

```
for i = 1 : n
    if notes (i) == 10
        nota1 = nota1 + 1;

    elseif notes (i) == 6.67
        nota2 = nota2 + 1;

    elseif notes (i) == 3.33
        nota3 = nota3 + 1;

    elseif notes (i) == 0
        nota4 = nota4 + 1;
    end
end
```

- Finalment es guarden les variables que contenen el resultat del compteig en una sola variable anomenada hnotes (referent a histograma de notes) i es grafiquen mitjançant la funció "bar", definint també els títols dels eixos:

```
hnotes = [nota1 nota2 nota3 nota4];
bar (hnotes)
xlabel('nota')
ylabel('freqüència')
```

Amb les notes simulades a mode d'exemple el primer histograma presenta el següent aspecte:

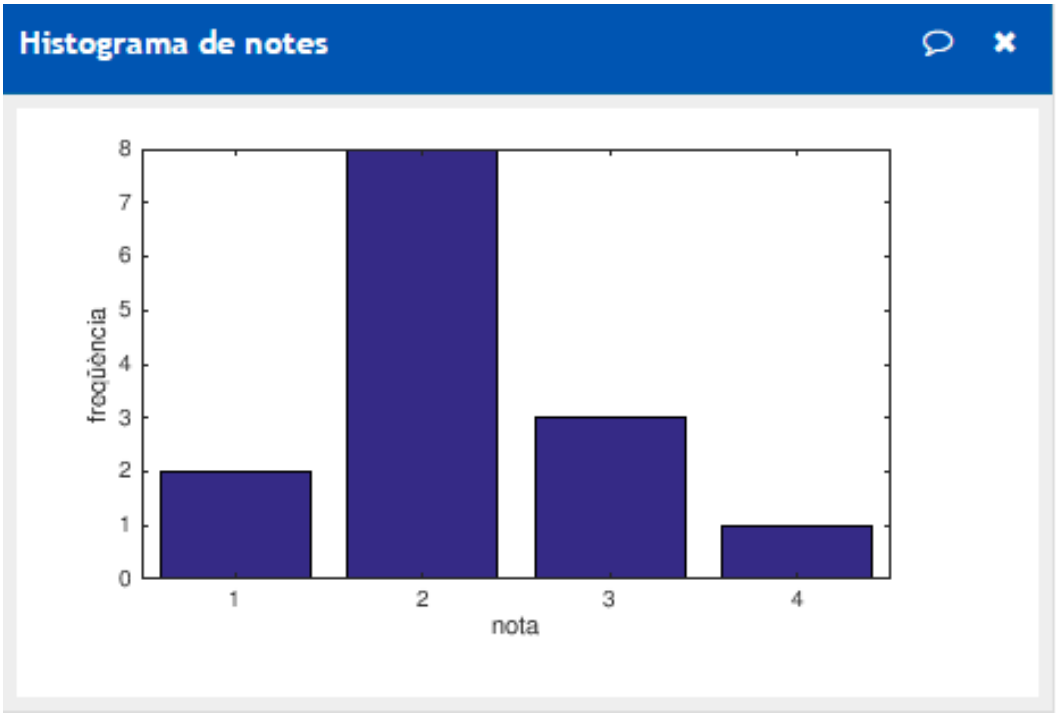

Figura 19: Histograma de notes. Font: Pròpia.

En el segon histograma es mostra la freqüència en que es repeteix cada error. El codi del segon histograma procedeix:

- Com en el cas del primer histograma es comença definint la ID del canal que es vol llegir (en aquest cas el 2) i la seva clau d'escriptura:

```
readChannelID = 109761;

readAPIKey = '8OOY9CG7FN5ELEAA';
```

- Seguidament s'assigna el contingut de cada camp del canal a una variable diferent corresponent a cada tipus d'error:

```
errorsa = thingSpeakRead(readChannelID, 'Field', 1, 'NumPoints', 30,
'ReadKey', readAPIKey);

errorsb = thingSpeakRead(readChannelID, 'Field', 2, 'NumPoints', 30,
'ReadKey', readAPIKey);

errorsc = thingSpeakRead(readChannelID, 'Field', 3, 'NumPoints', 30,
'ReadKey', readAPIKey);

errorsd = thingSpeakRead(readChannelID, 'Field', 4, 'NumPoints', 30,
'ReadKey', readAPIKey);

errorse = thingSpeakRead(readChannelID, 'Field', 5, 'NumPoints', 30,
'ReadKey', readAPIKey);

errorsf = thingSpeakRead(readChannelID, 'Field', 6, 'NumPoints', 30,
'ReadKey', readAPIKey);

errorsg = thingSpeakRead(readChannelID, 'Field', 7, 'NumPoints', 30,
'ReadKey', readAPIKey);
```

- Posteriorment es realitza la suma dels errors per separat per tal de disposar del nombre de vegades que s'ha comès cada error en variables diferents:

```
at=sum(errorsa);

bt=sum(errorsb);

ct=sum(errorsc);

dt=sum(errorsd);

et=sum(errorse);

ft=sum(errorsf);

gt=sum(errorsg);
```

- Finalment es guarden les variables que contenen les sumes dels diferents errors una sola variable errors i es grafiquen mitjançant la funció "bar": Com en el cas del primer histograma, a cada error li correspon un número d'identificació en l'eix horitzontal de l'histograma tal i com es mostra en la següent taula:

Taula 9: Correspondència entre el número d'identificació en l'eix horitzontal de l'histograma amb l'error. Font: Pròpia.

Error	Número d'identificació en l'eix horitzontal de l'histograma
ea	1
eb	2
ec	3
ed	4
ee	5
ef	6
eg	7

```
errors = [at bt ct dt et ft gt];

bar (errors);

xlabel('error')

ylabel('freqüència')
```

Finalment el segon histograma presenta el següent aspecte:

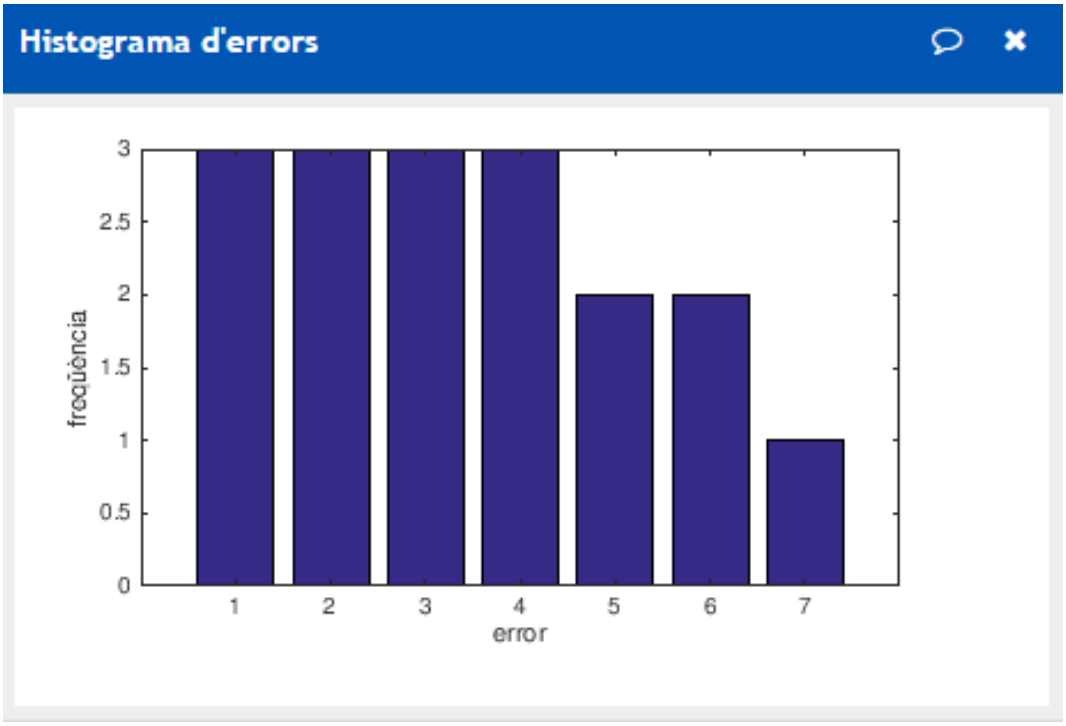

Figura 20: Histograma d'errors. Font: Pròpia.

Per altre banda, les eines d'anàlisi de dades permeten que, un cop identificat un nou error freqüent entre els alumnes, les dades generades en l'ús del programa puguin tornar a ser processades per tal de comprovar quants dels errors que havien estat classificats com a desconeguts són l'error identificat en qüestió.

Suposant un nou error del problema exemple anomenat "eh" que consisteixi en donar al corrent màssic "mP" el valor que s'esperaria que es donés al corrent màssic "mD", el codi per tornar a processar les dades és tal com:

- Es comença definint les IDs dels canals que es volen llegir i les claus de lectura, en aquest cas del canal 1 i del canal 3:

```
readChannelID1 = 109707;

readChannelID2 = 160636;



readAPIKey1 = '1ARLSTWMQ0BNMR9V';

readAPIKey2 = 'AY19GSSDB9SZ0D5C';
```

- Seguidament es llegeixen els camps necessaris per a fer els càlculs pertinents. A més de llegir el camp amb els DNIs per a la posterior tabulació de l'anàlisi, en el cas de l'error suposat "eh" per poder realitzar les comparacions és necessari disposar del valor correcte del corrent màssic "mD" calculat pel programa (pmD) i del valor introduït per l'usuari del corrent màssic "mP":

```
DNIs = thingSpeakRead(readChannelID1, 'Field', 1, 'NumPoints', 30, 'ReadKey', readAPIKey1);

pmD = thingSpeakRead(readChannelID2, 'Field', 2, 'NumPoints', 30, 'ReadKey', readAPIKey2);

mP = thingSpeakRead(readChannelID1, 'Field', 3, 'NumPoints', 30, 'ReadKey', readAPIKey1);
```

- Com que s'utilitzarà una estructura de bucle per llegir totes les entrades dels camps, s'assigna a la variable "n" mitjançant la funció "numel" la quantitat d'entrades emmagatzemades. Es defineix també la tolerància i les variables "DNI" i "errorh", on es guardarà el DNI i el resultat de la comparació de cada iteració respectivament:

```
n=numel (DNIs);

epsilon = 0.05;

DNI= 0;

errorh = 0;
```

- S'inicia el bucle des de 1 fins al número d'entrades comptades:

```
for i = 1 : n
```

- Al inici de cada iteració es reinicia el valor de la variable "eh", en la que se li assignarà el valor de 1 si la comparació dona un resultat positiu (s'ha comès l'error) o mantindrà el valor de 0 si la comparació dona un resultat negatiu:

```
    eh = 0;
```

- Per tal de realitzar la comparació s'assigna a la variable "pmP2" el valor que prendrà el corrent "mP" en cas que es cometi l'error en qüestió, que no és més que el valor de la resposta correcta calculada pel programa pel corrent màssic "mD" de la iteració en curs (pmD(i)).

```
pmP2 = pmD(i);
```

- Seguidament es calcula la diferència absoluta entre el valor de l'error i la resposta introduïda per l'usuari:

```
difmP2 = abs(pmP2-mP(i));
```

- En cas que la diferència sigui inferior al producte de la tolerància pel valor de l'error, es considera que s'ha comès l'error i s'assigna al a variable "eh" el valor 1 conforme l'error s'ha comès:

```
if difmP2< epsilon*pmP2


  eh =  1;
```

- A continuació comença un condicional per tal d'adequar el format de les variables que es tabularan al final. En cas que la variable "DNI" tingui un valor de 0 significa que és la primera iteració i la variable pren el valor del DNI de la primera entrada. Per la variable "errorh" que conté el resultat de les comparacions dels errors el procediment és el mateix:

```
 if DNI == 0


    DNI = DNIs (i);
    errorh = eh;
```

- En cas que no sigui la primera iteració, s'afegeix el DNI i el resultat de la comparació de l'error a les variables corresponents:

```
 else


   DNI = [DNI;DNIs(i)];

   errorh = [errorh; eh];


 end
```

- Es generà també un segon condicional que realitza la mateixa funció però en cas de que la comparació amb l'error sigui negativa i s'acaben els condicionals i el bucle:

```
else


  if DNI == 0


    DNI = DNIs (i);

    errorh = eh;


  else


    DNI = [DNI;DNIs(i)];

    errorh = [errorh; eh];


  end
 end
end
```

- S'utilitza el comandament "format long" per tal que les dades es tabulin sense decimals i s'utilitza la funció "table" per obtenir una taula amb els resultats de l'anàlisi del nou error identificat:

```
format long

table (DNI, errorh)
```

La taula en qüestió per les mateixes entrades simulades mostra el següent aspecte:

Taula 10: Taula amb resultats de l'anàlisi de dades del problema 1. Font: Pròpia.

DNI	errorh
39395908	0
39395908	0
39395908	0
39395908	0
39395908	0
39395908	0
39395908	0
39395908	0
39395908	0
39395908	0
39395908	0
39395908	0
39395908	0
39395908	1

En el cas de les notes simulades només la última entrada ha comès l'error analitzat.

5. Disseny d'un problema

L'arxiu del problema tal i com es comenta en la introducció al programa està programat en Matlab i la seva interfície creada amb GUIDE (graphical user interface development enviornment).

GUIDE és una eina de Matlab per crear aplicacions de software que permeten automatitzar tasques i càlculs. GUIDE disposa de controls tals com menús, barres d'eines i botons entre d'altres.

En aquest apartat es planteja un segon problema exemple i se'n aborda el seu disseny des de l'anàlisi del problema per transformar-lo en un codi executable i en una interfície interactuable fins a l'establiment de connexió amb la base de dades i el procés de tractament de dades.

5.1 Problema exemple 2

El problema que es planteja està extret de l'assignatura d'operacions de separació. L'enunciat del problema es mostra a continuació.

Enunciat Problema exemple 2:

- En un evaporador senzill es vol concentrar una dissolució aquosa des del 10% al 30% en pes. La dissolució no presenta augment en el punt d'ebullició. Per a la calefacció es disposa de vapor viu a 3.6 atm de pressió absoluta, i la pressió absoluta a la cambra d'evaporació és de 400 mm Hg. La dissolució diluïda es troba a 18ºC.

  Per tal de no perdre tanta quantitat de calor es disposa d'un sol bescanviador de calor de doble tub a contracorrent per preescalfar la dissolució diluïda abans de la seva entrada a l'evaporador. Per aquest preescalfament es dubta entre fer servir com a líquid de calefacció la dissolució concentrada que surt de l'evaporador, o bé l'aigua condensada que surt de la cambra de calefacció, ja que l'evaporador és de poc consum i l'aigua no es recircula a la caldera.

Demostrar mitjançant els balanços quina és la millor opció. Per això caldrà que doneu resposta a :

1) Temperatura d'entrada de l'aliment a l'evaporador si el preescalfament es fa mitjançant la calor de la dissolució concentrada (tfb1).

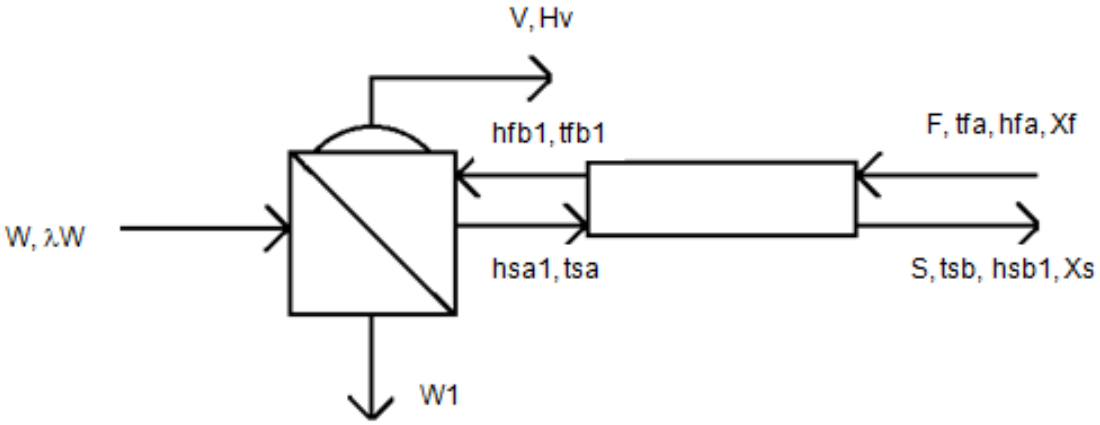

Figura 21: Esquema del sistema. Qüestió 1 problema exemple 2. Font: Pròpia.

2) Temperatura d'entrada de l'aliment a l'evaporador si el preescalfament es fa mitjançant l'aigua condensada a la cambra de calefacció (tfb2).

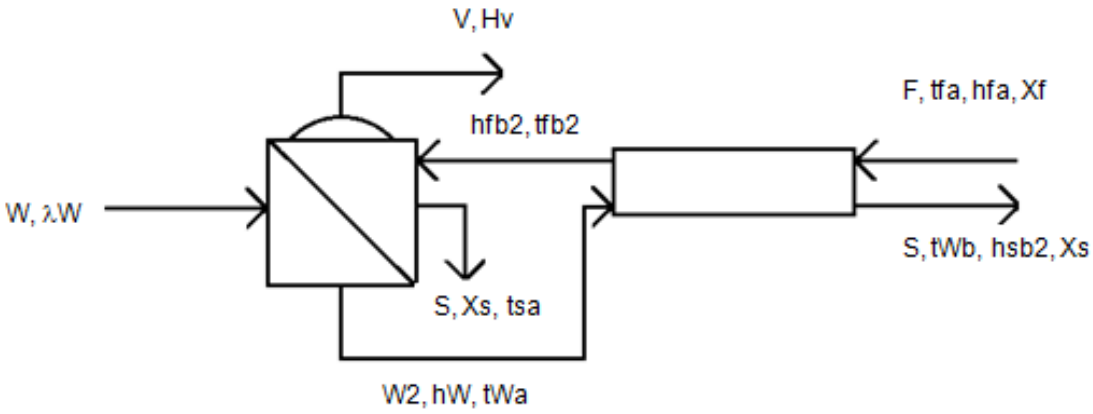

Figura 22: Esquema del sistema. Qüestió 2 problema exemple 2. Font: Pròpia.

3) Quina és la millor opció?

Dades :

F= 1 kg/h

CpF = CpS= 1 kcal/kgºC

(UoAo)bescanviador = 1 kcal/hºC

En aquest segon problema exemple, els paràmetres personalitzats que variaran en funció del DNI de l'usuari són:

Taula 11: Paràmatres problema exemple 2. Font: Pròpia.

Paràmatre	Valor original del problema	Valor que pren amb el programa
Xf	0,1	0,1 + DNI(1)/100
Xs	0,3	0,3 + DNI(2)/100
tfa	18	18 + DNI(3)

Per tal de comprovar l'estabilitat del problema amb els rangs de valors donats als paràmetres es planteja el procés de resolució.

Per resoldre el problema, són necessàries dades addicionals a les aportades en l'enunciat. Del recull de dades de la mateixa assignatura se'n extreuen:

W a 3,6 atm $\longrightarrow$ tWa = 139,18 ºC , λW = (652,2-139,8) Kcal/kg

Pressió del vapor a 400 mmHg $\longrightarrow$ teb = tsa = 83 ºC , Hv = 632,42 Kcal/Kg

Partint d'un cabal màssic d'aliment de 1kg/h es plantegen les equacions per resoldre el primer apartat:

- Balanços de matèria del sistema:

  $F = S + V$

  $Xf = S \cdot Xs$

  On el cabal F és ignorat ja que el seu valor és 1.

- Balanç entàlpic del sistema:

  W1·(652.2-139.8)+tfb1 = S·83+V·632.42

- Balanços d'energia en l'intercanviador de calor:

  q1 = ((tsb-tfa)-(83-tfb1))/log((tsb-tfa)/(83-tfb1))

  q1 = (tfb1-tfa)

  q1 = S·(83-tsb)

Es plantegen també les equacions pel segon apartat:

- Balanç entàlpic del sistema:

  tfb2+W2·(652.2-139.8) = S·83+V·632.42

- Balanços d'energia en l'intercanviador de calor:

  q2 = ((139.18-tfb2)-(tWb-tfa))/log((139.18-tfb2)/(tWb-tfa))

  q2 = tfb2-tfa

  q2 = W2*(139.18-hW)

- Entalpia del vapor condensat:

  hW = tWb·1.02

I les equacions per les economies del tercer apartat:

E1=V/W1

E2=V/W2

Un cop plantejades les equacions, s'escriuen en llenguatge de Matlab:

```
syms V S tsb tfb1 q1 W1 hW tfb2 q2 W2 tWb

%Balanços de matèria del sistema
eq1 = 1 == S + V;
eq2 = Xf == S * Xs;

%Balanç entàlpic del sistema1
eq3 = W1*(652.2-139.8) +tfb1 == S*83+V*632.42;

%Balanços d'energia en l'intercanviador de calor1
eq4 = q1 == ((tsb-tfa)-(83-tfb1))/log((tsb-tfa)/(83-tfb1));
eq5 = q1 == (tfb1-tfa);
eq6 = q1 == S*(83-tsb);

%Balanç entàlpic del sistema2
eq7 = tfb2+W2*(652.2-139.8) == S*83+V*632.42;

%Balanços d'energia en l'intercanviador de calor2
eq8 = q2 == ((139.18-tfb2)-(tWb-tfa))/log((139.18-tfb2)/(tWb-tfa));
eq9 = q2 == tfb2-tfa;
eq10 = q2 == W2*(139.18-hW);

%Entalpia del vapor condensat
eq11 = hW == tWb*1.02;



sol = solve([eq1, eq2, eq3, eq4, eq5, eq6, eq7, eq8, eq9, eq10,eq11],
[V, S, tsb, tfb1, q1, W1, hW, tfb2, q2, W2, tWb]);
```

```
V = double(sol.V);
S = double(sol.S);
tsb1 = double(sol.tsb1);
tfb1 = double(sol.tfb1);
q1 = double(sol.q1);
W1 = double (sol.W1);
hW = double(sol.hW);
tfb2 = double(sol.tfb2);
q2 = double(sol.q2);
W2 = double(sol.W2);
tWb = double(sol.tWb);

E1=V/W1;
E2=V/W2;

if E1 > E2
    E=1
else
    E=2

end
```

On "E" prendrà el valor de 1 si la opció més econòmica és la de l'apartat 1 o 2 en cas que l'opció més econòmica sigui l'apartat 2.

Un cop es disposa del càlcul automatitzat del problema es procedeix a estudiar com varien els valors de les respostes amb totes les combinacions possibles de valors dels paràmetres en els casos extrems.

Els valors extrems que poden prendre els paràmetres són:

Taula 12: Valors extrems dels paràmatres en el problema exemple 2. Font: Pròpia.

			Limit del rang	
Paràmetre	**Valor original del problema**	**Valor que pren amb el programa**	**Màxim (Xifra de DNI 9)**	**Mínim (Xifra de DNI 0)**
Xf	0,1	0,1 + DNI(1)/100	0,19	0,1
Xs	0,3	0,3 + DNI(2)/100	0,39	0,3
tfa	18 (ºC)	18 + DNI(3) (ºC)	27 (ºC)	18 (ºC)

Es plantegen els casos de combinacions d'extrems, juntament amb el DNI corresponent per generar els paràmetres del cas i els resultats obtinguts:

Taula 13: Casos de combinacions d'extrems en el problema exemple 2. Font: Pròpia.

	Limits del rang dels paràmetres			**DNI**	**Valors de les respostes**		
Cas	**Xf**	**Xs**	**tfa (ºC)**		**tfb1 (ºC)**	**tfb2 (ºC)**	**E**
Cas 1	MIN	MIN	MIN	00000000	37,62	72,97	2
Cas 2	MIN	MIN	MAX	00900000	43,90	77,61	2
Cas 3	MIN	MAX	MAX	09900000	40,76	79,38	2
Cas 4	MAX	MIN	MAX	90900000	51,17	66,51	2
Cas 5	MAX	MAX	MIN	99000000	42,84	68,07	2
Cas 6	MAX	MIN	MIN	90000000	46,05	61,32	2
Cas 7	MAX	MAX	MAX	99900000	48,40	72,96	2

Com es pot observar la variació dels valors dels paràmetres comporta canvis lleugers en les respostes mantenint la coherència del problema.

5.2 Interfície

Un cop plantejat el problema i comprovada la seva estabilitat, es procedeix a dissenyar la interfície.

S'accedeix a GUIDE des de Matlab clicant a "New" i posteriorment a "Graphical User Interface" tal i com es mostra a continuació

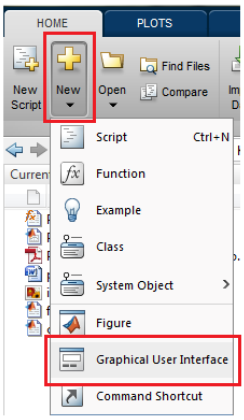

Figura 23: Accés a GUIDE. Font: Matlab.

En la següent pantalla que es mostra es selecciona "Blank GUI (Default)" i es clica "OK" :

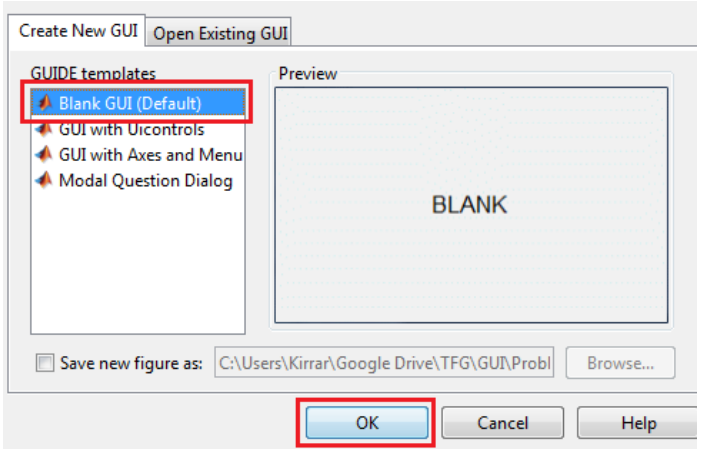

Figura 24: Creació d'interfície en blanc. Font: Matlab.

La pròxima pantalla que es mostra és la de disseny de la interfície. Situats en una barra d'eines a l'esquerra s'hi trobem els elements que s'utilitzen per dissenyar-la:

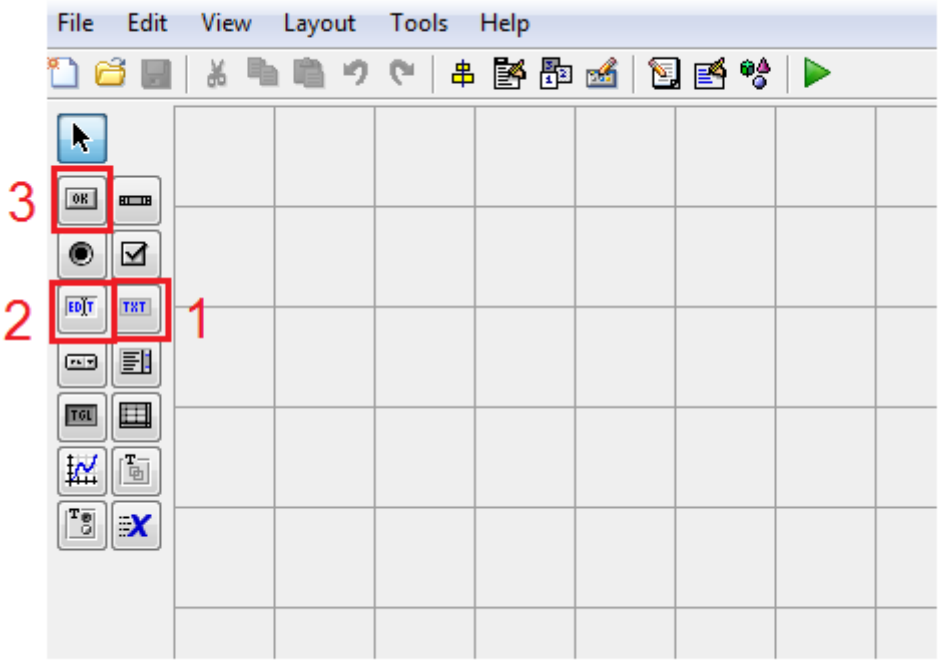

Figura 25: Opcions de la barra d'eines emprades. Font: GUIDE, Matlab.

On:

- 1: Caselles de text no editable.
- 2: Caselles de text editables.
- 3: Botons.

Primerament es creen les caselles de text no editable. Aquests s'utilitzen per marcar les seccions dels diferents estadis d'ús del programa (generació de paràmetres, avaluació i enviament de resultat). S'utilitzen també per assignar-hi els missatges d'errors en l'ús del programa, el nom dels paràmetres, el seu valor generat i les unitats corresponents, el nom de les variables de resposta, la nota i els comentaris corresponents a cada qüestió.

En el cas del problema exemple 2 es generen les caselles de text no editable amb les dimensions en files per columnes tal i com es mostra a continuació:

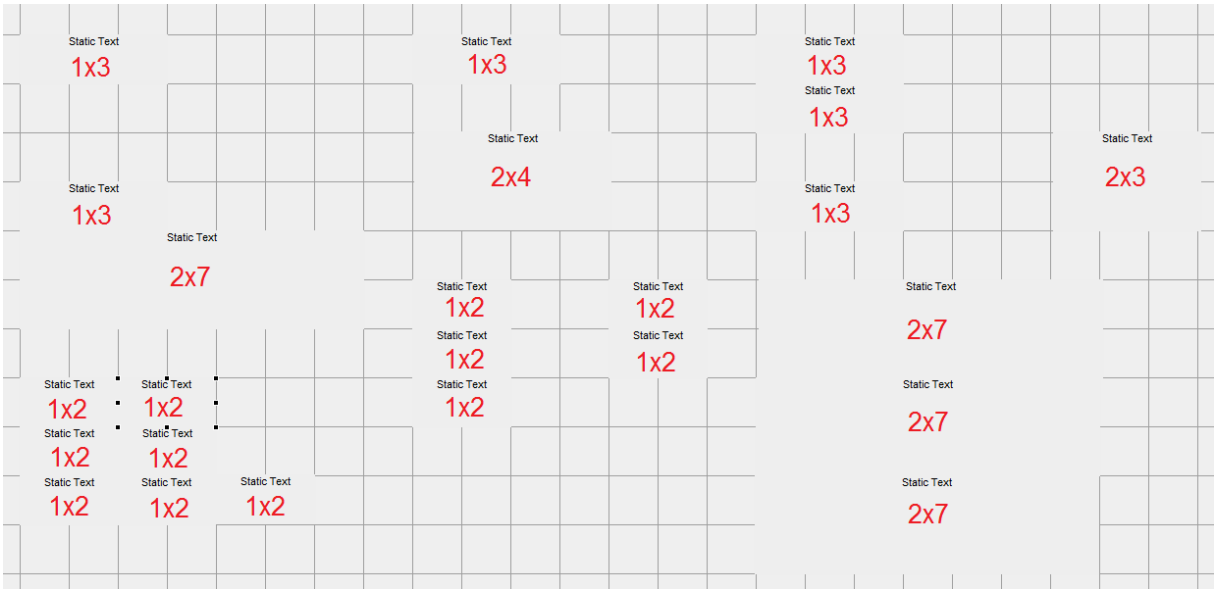

Figura 26: Distribució de caselles de text no editable. Font: Pròpia.

Un cop generades les caselles de text no editable, fent-hi clic dret s'accedeix a propietats:

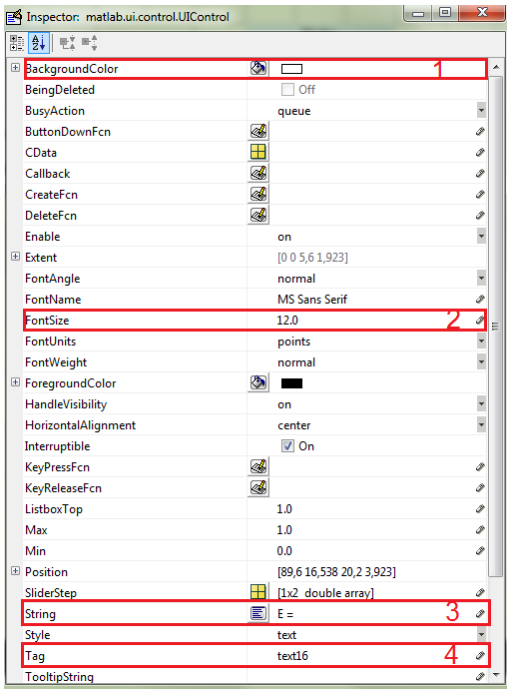

Figura 27: Panell de propietats d'una casella de text no editable. Font: Pròpia.

On seguint la numeració de la figura 27 es canvia:

1. El color.
2. La mida de la font.
3. El text que conté.
4. L'etiqueta o "tag" (és el nom amb el qual es crida l'element en el codi).

Es mostra a continuació l'estat de la interfície un cop realitzats els canvis pertinents, així com els "tags" que es canvien (només dels elements que hauran de ser cridats en l'execució del codi):

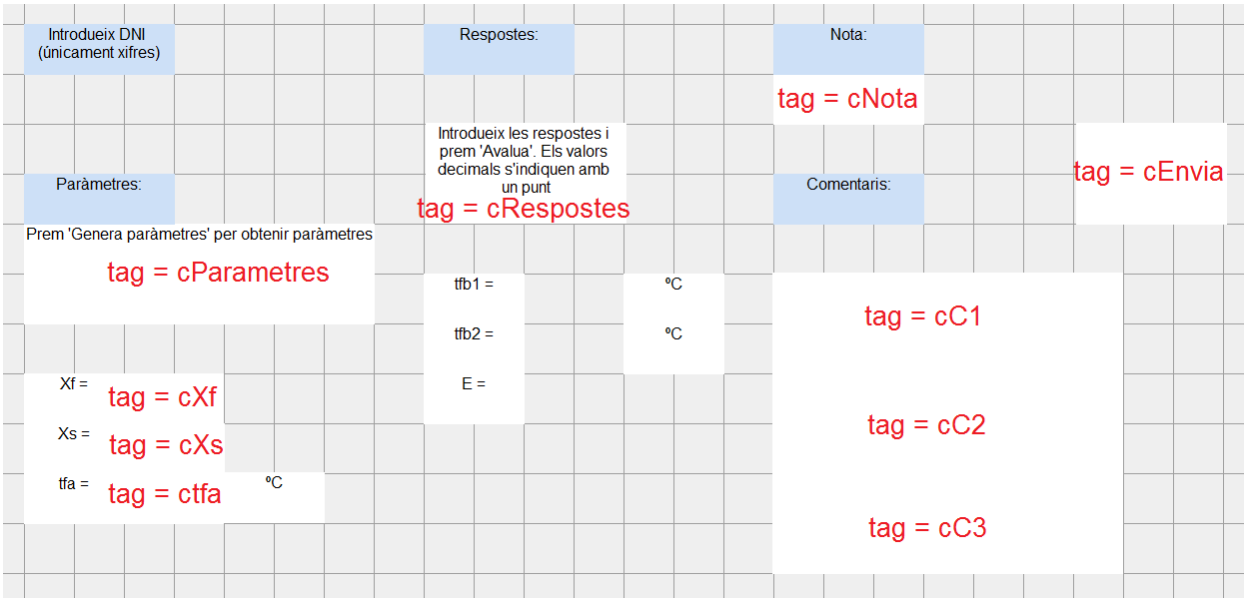

Figura 28: Caselles de text no editables amb format ajustat i "tags". Font: Pròpia.

Seguidament es creen els panells de text editable, que s'utilitzen per permetre a l'usuari introduir el DNI i les respostes. Un cop creats els panells de text editable i editat el seu format, la interfície presenta l'aspecte que es mostra a continuació, juntament amb el "tag" assignat:

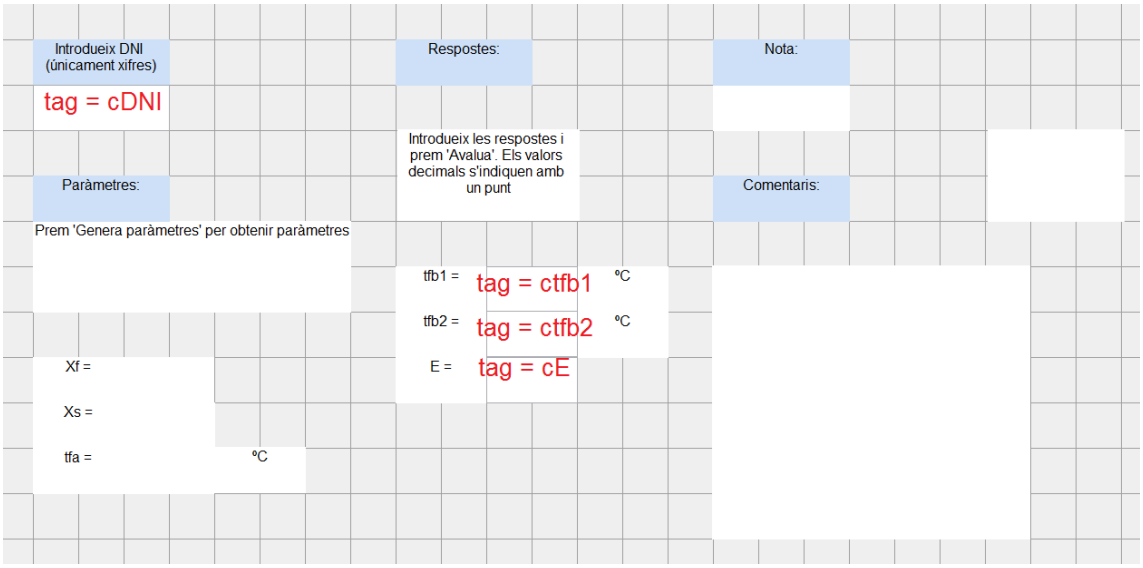

Figura 29: Caselles de text editables i "tags". Font: Pròpia.

Per últim es creen els botons, que un cop adequat el seu format, la interfície presenta l'aspecte que es mostra tot seguit, juntament amb el "tag" assignat:

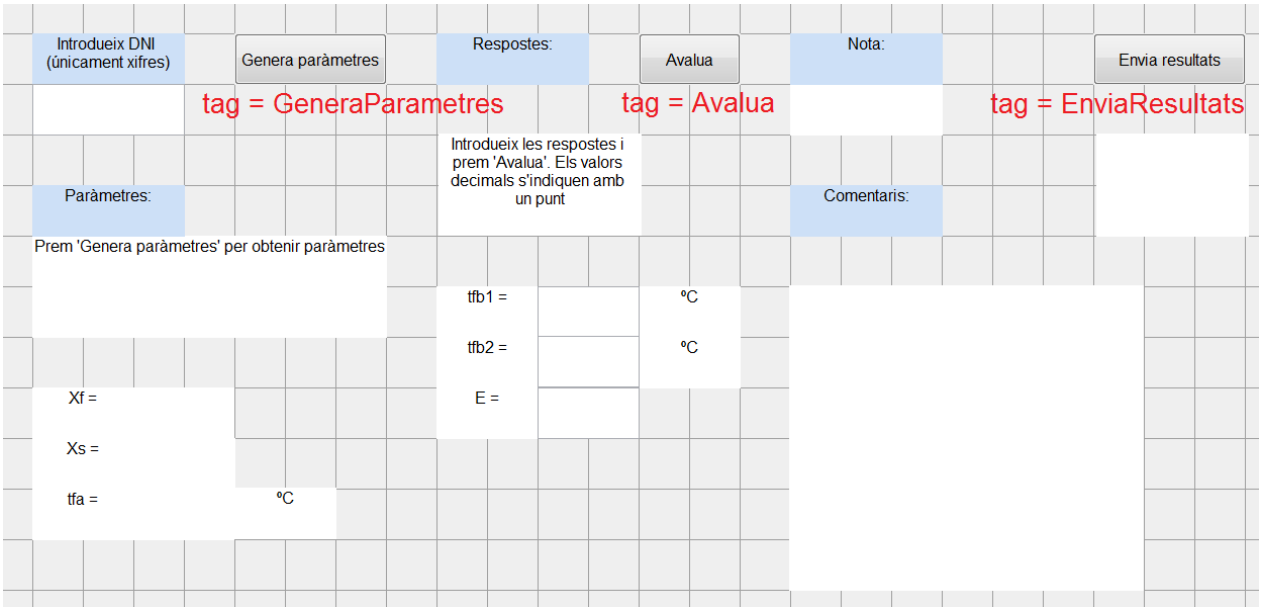

Figura 30: Botons i "tags". Font: Pròpia.

Un cop generats els botons finalitza el disseny de la interfície. Es procedeix a guardar la interfície amb el nom "ProblemaExemple2". Es guarda un fitxer d'extensió ".fig" que conté la interfície i alhora es genera un fitxer d'extensió ".m" que conté el codi de la interfície dissenyada.

És en el fitxer de codi generat on es continua el disseny del problema.

5.3 Codi

Primerament en obrir el fitxer del codi es fa neteja de totes les línies de comentaris (indicades pel seu color verd i el símbol "%" al inici de la línia) i un cop feta la neteja, es procedeix a dissenyar el codi de la funció obertura.

5.3.1 Funció "Funció obertura"

La funció obertura presenta exactament el mateix codi que la funció obertura del problema exemple, ja que simplement es defineixen dues variables de comprovació:

```
function ProblemaExemple2_OpeningFcn(hObject, eventdata, handles,
varargin)

handles.output = hObject;

handles.parametres=0;
handles.avaluat=0;

guidata(hObject, handles);
```

5.3.2 Funció "Genera paràmetres"

Es continua el disseny del codi amb les funcions dels botons.

El primer que es dissenya és el codi del botó "Genera paràmetres", que en estructura és idèntic que el de la mateixa funció pel problema exemple, simplement cal canviar els "tags" dels paràmetres i les caselles corresponents i ajustar el seu càlcul.

Es mostra a continuació el codi amb els canvis ressaltats amb groc del botó "Genera paràmetres":

```
function Genera_Parametres_Callback(hObject, eventdata, handles)

if handles.avaluat == 1

    set (handles.cParametres, 'String', 'Ja has estat avaluat amb un
DNI. Per canviar de DNI reinicia el programa');

else

    handles.DNI = get (handles.cDNI, 'String');

    if length(handles.DNI) == 0

        set (handles.cParametres, 'String', 'Error al calcular els
paràmetres, introdueix DNI')
```

- Es canvien els "tags" dels paràmetres:

```
        set (handles.cXf, 'String', '');
        set (handles.cXs, 'String', '');
        set (handles.cE, 'String', '');

elseif length(handles.DNI) == 8

        DNIv = size (str2num (handles.DNI(1)))+size (str2num
(handles.DNI(2)))+size (str2num (handles.DNI(3)))+size (str2num
(handles.DNI(4)))+size (str2num (handles.DNI(5)))+size (str2num
(handles.DNI(6)))+size (str2num (handles.DNI(7)))+size (str2num
(handles.DNI(8)));
```

```
if DNIv == 8
```

– S'ajusta el seu càlcul i es canvien també els "tags" de les caselles:

```
            handles.Xf = (str2num(handles.DNI(1)))/100 + 0.1;
            handles.Xs = (str2num(handles.DNI(2)))/100 + 0.3;
            handles.tfa = str2num(handles.DNI(3)) + 18;

            set (handles.cXf, 'String', handles.Xf);
            set (handles.cXs, 'String', handles.Xs);
            set (handles.ctfa, 'String', handles.tfa);

            set (handles.cParametres, 'String', 'Paràmatres calculats
amb èxit')

            handles.parametres = 1;

        else

            set (handles.cParametres, 'String', 'Error al calcular els
paràmetres, revisa el DNI. Ha de ser de 8 xifres i no contenir
lletres')
            set (handles.cXf, 'String', '');
            set (handles.cXs, 'String', '');
            set (handles.cE, 'String', '');

        end
```

```
    else

        set (handles.cParametres, 'String', 'Error al calcular els
paràmetres, revisa el DNI. Ha de ser de 8 xifres i no contenir
lletres')
        set (handles.cXf, 'String', '');
        set (handles.cXs, 'String', '');
        set (handles.cE, 'String', '');

    end

    guidata(hObject, handles);

end
```

5.3.3 Funció "Avalua"

El següent pas és dissenyar el codi del botó "Avalua". De nou, l'estructura és la mateixa que pel problema exemple.

Es mostra a continuació el codi amb comentaris sobre els canvis del botó "Avalua":

```
function Avalua_Callback(hObject, eventdata, handles)
```

- Es defineixen les variables dels errors i de la nota. Les variables dels errors es correspondran amb els errors comesos tal i com es mostra en la següent taula:

Taula 14: Taula d'equivalències d'errors problema exemple 2. Font: Pròpia.

	Variables d'errors					
	ea	**eb**	**ec**	**ed**	**ee**	**ef**
Tipus d'error	Resposta buida qüestió 1	Resposta buida qüestió 2	Resposta buida qüestió 3	Error desconegut qüestió 1	Error desconegut qüestió 2	Error desconegut qüestió 3

El nomenament dels errors continuaria pels errors freqüents que es vagin identificant seguint el mateix format (eg, eh...)

```
handles.nota=0;
handles.ea = 0;
handles.eb = 0;
handles.ec = 0;
handles.ed = 0;
handles.ee = 0;
handles.ef = 0;
```

```
if handles.parametres == 1
```

– Es canvien els noms de les variables amb les respostes de l'usuari i dels "tags" corresponents en la seva crida:

```
handles.tfb1 = str2num(get (handles.ctfb1, 'String'));
handles.tfb2 = str2num(get (handles.ctfb2, 'String'));
handles.E = str2num(get (handles.cE, 'String'));
```

– El codi on és calculen les respostes del problema canvia tot sencer. És el mateix que s'ha dissenyat ja anteriorment per comprovar l'estabilitat del problema, simplement cal afegir l'estructura "handles" en la definició de les variables que contenen les respostes i les dels paràmatres i afegir una "p" a les variables que calcularà el programa que tenen el mateix nom que les variables respostes introduïdes per l'usuari (ptfb1, ptfb2, pE):

```
syms V S tsb ptfb1 q1 W1 hW ptfb2 q2 W2 tWb

eq1 = 1 == S + V;
eq2 = handles.Xf == S * handles.Xs;

eq3 = W1*(652.2-139.8) +ptfb1 == S*83+V*632.42;

eq4 = q1 == ((tsb-handles.tfa)-(83-ptfb1))/log((tsb-
handles.tfa)/(83-ptfb1));
eq5 = q1 == (ptfb1-handles.tfa);
eq6 = q1 == S*(83-tsb);

eq7 = ptfb2+W2*(652.2-139.8) == S*83+V*632.42;

eq8 = q2 == ((139.18-ptfb2)-(tWb-handles.tfa))/log((139.18-
ptfb2)/(tWb-handles.tfa));
eq9 = q2 == ptfb2-handles.tfa;
eq10 = q2 == W2*(139.18-hW);

eq11 = hW == tWb*1.02;
```

```
    sol = solve([eq1, eq2, eq3, eq4, eq5, eq6, eq7, eq8, eq9,
eq10,eq11], [V, S, tsb, ptfb1, q1, W1, hW, ptfb2, q2, W2, tWb]);


    handles.V = double(sol.V);
    handles.S = double(sol.S);
    handles.tsb = double(sol.tsb);
    handles.ptfb1 = double(sol.ptfb1);
    handles.q1 = double(sol.q1);
    handles.W1 = double (sol.W1);
    handles.hW = double(sol.hW);
    handles.ptfb2 = double(sol.ptfb2);
    handles.q2 = double(sol.q2);
    handles.W2 = double(sol.W2);
    handles.tWb = double(sol.tWb);


    E1 = handles.V/handles.W1;
    E2 = handles.V/handles.W2;


    if E1 > E2


      handles.pE=1;


    else


      handles.pE=2;


    end
```

- En cas de disposar d'errors freqüents seria el moment de calcular el valor que prendria la variable de resposta per l'error comès.

- Es canvien a continuació els noms de les variables en el pas de calcular la diferència entre les respostes de l'usuari i les calculades pel programa. Cal tenir en compte que en la qüestió 3 no s'introdueix valor, sinó un 1 o un 2 en funció de quina sigui la opció més econòmica, per tant, no es precisa de calcular la seva diferència:

```
diftfb1 = abs(handles.ptfb1-handles.tfb1);
diftfb2 = abs(handles.ptfb2-handles.tfb2);
```

- Es canvien també els noms en les variables de comprovació de caselles de resposta buides:

```
tfb1empty = isempty (get (handles.ctfb1, 'String'));
tfb2empty = isempty (get (handles.ctfb2, 'String'));
Eempty = isempty (get (handles.cE, 'String'));

epsilon = 0.05;
```

- A continuació comença el condicional de comparació de respostes. En estructura és idèntic al del problema exemple, però caldrà canviar els noms de les variables per les corresponents del problema i adequar la part del condicional de la qüestió 3, ja que no es realitza comparacions entre diferencies de valors:

```
if diftfb1<epsilon*handles.ptfb1
    C1  = 'Qüestió 1: Resposta correcta';
    handles.nota = handles.nota+1;

elseif tfb1empty == 1
    C1= 'Qüestió 1: Error. Introdueix una resposta';
    handles.ea = 1;
```

```
else

    C1 = 'Qüestió 1: Resposta incorrecta. Error desconegut';

     handles.ed = 1;

end


if diftfb2<epsilon*handles.ptfb2

    C2 ='Qüestió 2: Resposta correcta';

    handles.nota = handles.nota+1;

elseif tfb2empty == 1

    C2 = 'Qüestió 2: Error. Introdueix una resposta';

    handles.eb = 1;

else

    C2 ='Qüestió 2: Resposta incorrecta. Error desconegut';

    handles.ee = 1;

end
```

- La següent part del condicional canvia lleugerament ja que en comptes de comparar amb una diferència de valors, es comparà amb la opció introduïda (1 o 2):

```
if handles.E == handles.pE

    C3  ='Qüestió 3: Resposta correcta';

     handles.nota = handles.nota+1;


elseif Eempty == 1

    C3 = 'Qüestió 3: Error. Introdueix una resposta';

    handles.ec = 1;


else

    C3 ='Qüestió 3: Resposta incorrecta. Error desconegut';

    handles.ef = 1;

end


    handles.nota = 10*handles.nota/3;
    handles.nota = round(handles.nota, 2);


    handles.avaluat = 1;


    guidata(hObject, handles);


    set (handles.cNota, 'String', handles.nota)
    set (handles.cC1, 'String', C1)
    set (handles.cC2, 'String', C2)
    set (handles.cC3, 'String', C3)
    set (handles.cRespostes, 'String', 'Avaluació amb èxit')

else


    set (handles.cRespostes, 'String', 'No es disposa de paràmatres
per a poder avaluar')


end
```

5.3.4 Canals de ThingSpeak

Per dissenyar el codi del botó restant "Envia resultats", primerament és necessari crear els canals de ThingSpeak per tal de disposar de les claus d'escriptura.

Cal doncs decidir quantes variables es desitja emmagatzemar per saber quants canals han de ser creats (partint de que cada canal conté 8 camps).

El DNI, la nota i les respostes de l'usuari es guardaran en un primer canal. Els errors, en un segon canal i les respostes calculades pel programa en un tercer canal.

Un cop decidit quants canals són necessaris, es procedeix a crear els canals.

Per tal d'accedir i utilitzar ThingSpeak el primer que cal fer és crear un compte. Per fer-ho, cal accedir a "https://thingspeak.com/" i clicar a "Sign up", indicat per un requadre vermell en la imatge que es mostra a continuació:

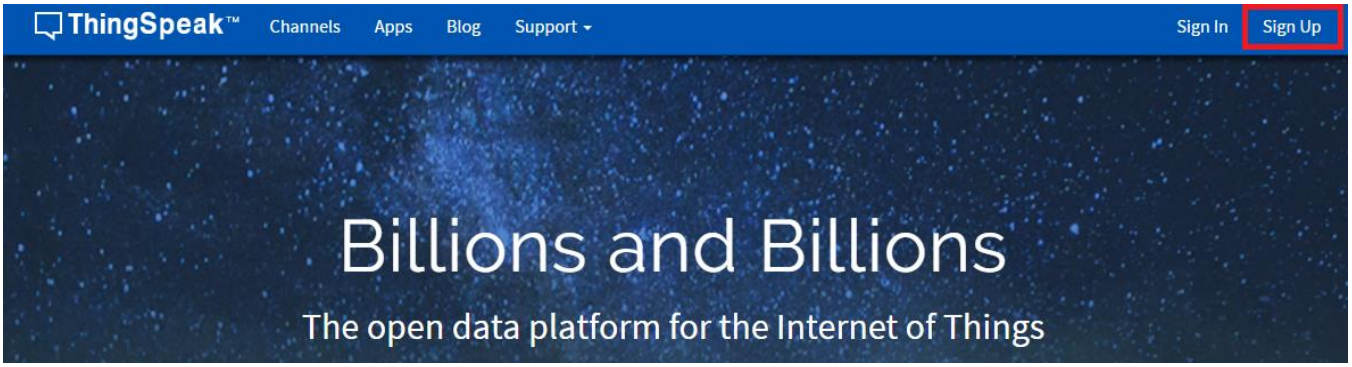

Figura 31: Portada de ThingSpeak. Font: https://thingspeak.com/

A continuació cal omplir el requadres que es mostren en la imatge a continuació i clicar a "Create Account":

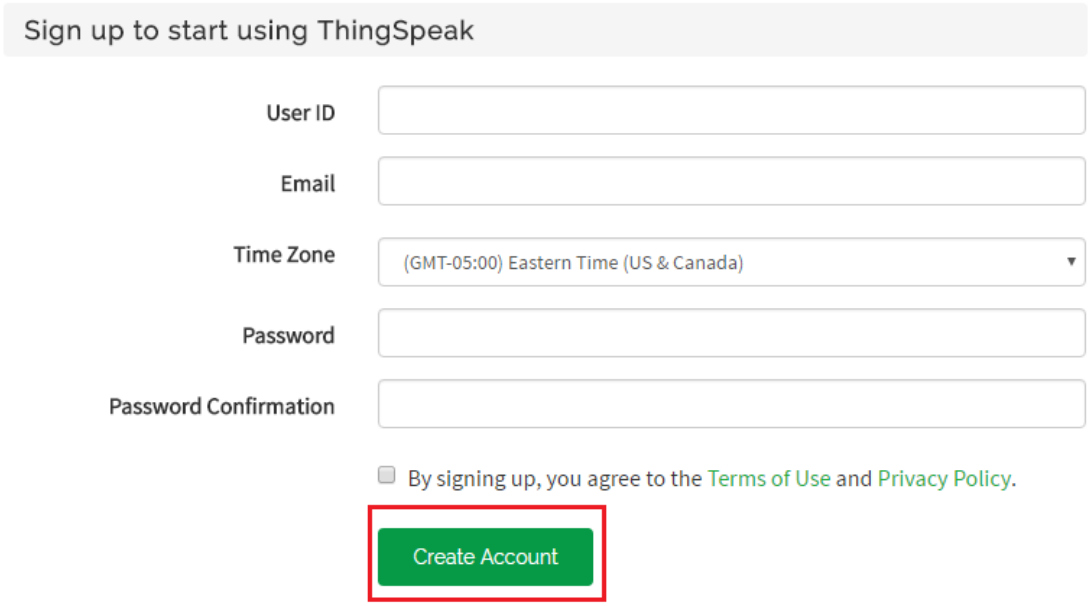

Figura 32: Creació de compte de ThingSpeak. Font: https://thingspeak.com/

Seguidament es mostra la següent pàgina on clicant a "New Channel" es procedeix a crear el primer canal:

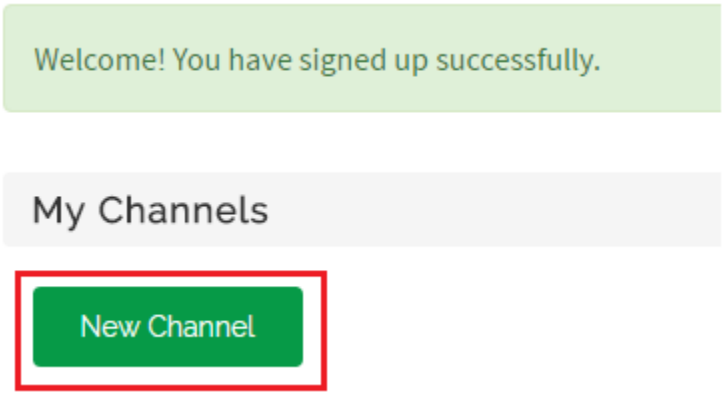

Figura 33: Panell de creació de canal. Font: https://thingspeak.com/.

En la pantalla de creació del nou canal es demana un nom pel canal i es pot escriure també una breu descripció d'aquest:

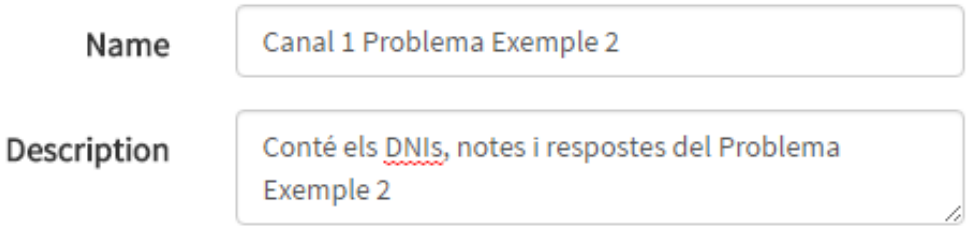

Figura 34: Nom i descripció del canal. Font: Pròpia.

Hi segueixen els 8 camps, així com un requadre al seu costat que permet activar-los o desactivar-los:

Figura 35: Camps del canal 1 problema exemple 2. Font: Pròpia.

Existeix també la opció de fer el canal públic. Cal assegurar-se de que la pestanya no queda marcada:

Figura 36: Casella per convertir el canal en públic. Font: https://thingspeak.com/.

Finalment es clica "Save Channel" al final de la pàgina i es crea el canal.

Se'ns mostra llavors la vista privada del canal:

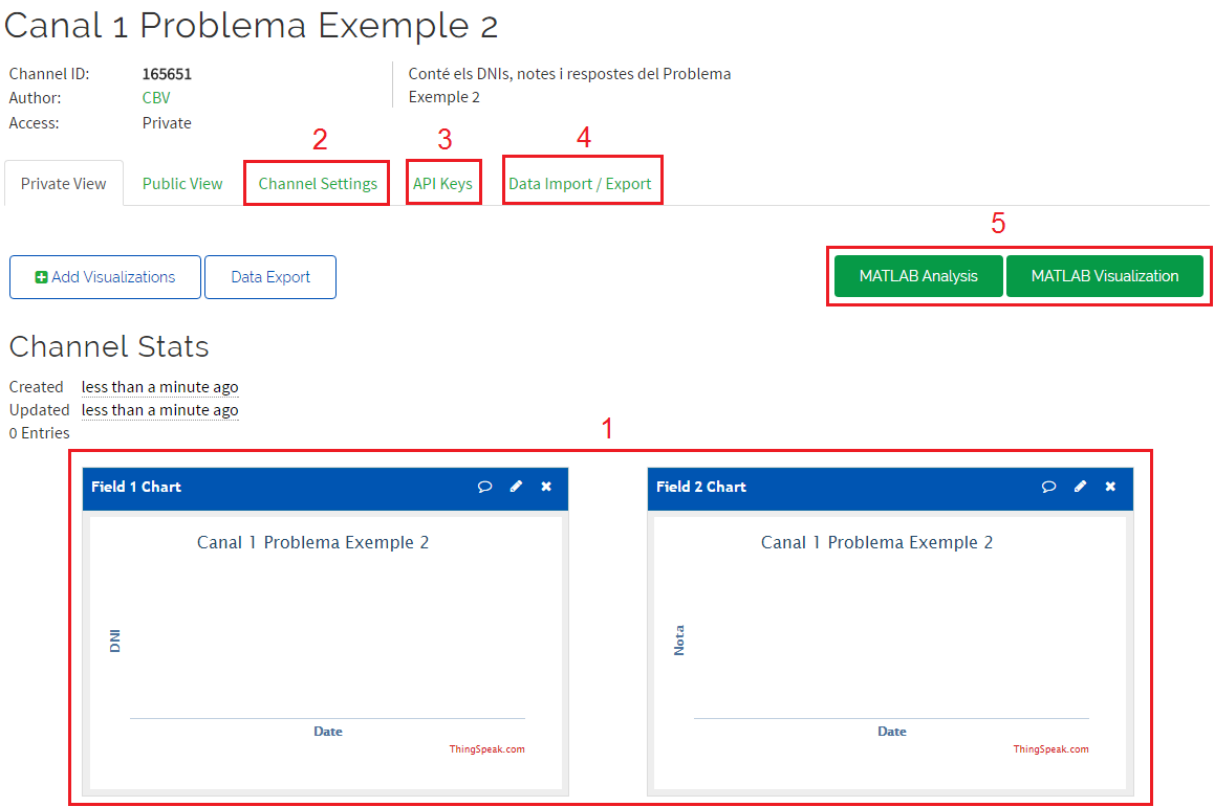

Figura 37: Vista privada del canal. Font: https://thingspeak.com/.

On seguint la numeració que s'indica en la figura 37:

1. És la zona on es mostren les visualitzacions generades. Al crear el canal se'n generen per defecte i és necessari eliminar-les utilitzant la creueta de la part superior dreta de la visualització corresponent.
2. Pestanya d'ajustos del canal. Té el mateix aspecte que el que es mostra anteriorment en les figures de creació del canal.
3. Claus API. És en aquesta secció d'on s'obtindràn les claus d'escriptura i lectura dels canals.
4. Importació i exportació de dades. S'accedirà a aquesta secció quan es desitgi descarregar les dades en format CSV.
5. Opcions d'anàlisis i visualitzacions.

Per tal de continuar amb la creació dels dos canals restants, es clica a "Channels" a la part superior de la pàgina i seguidament a "My channels":

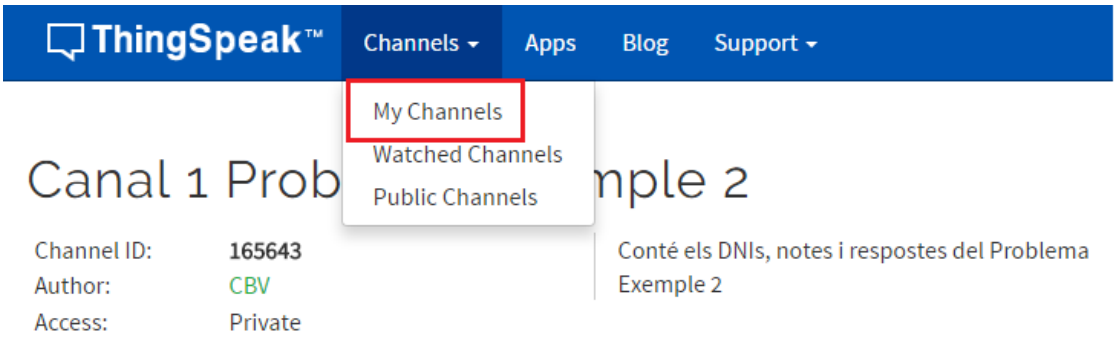

Figura 38: Accés al llistat de canals. Font: https://thingspeak.com/.

En aquest apartat de la pàgina es mostra el llistat de canals que conté el compte de ThingSpeak.:

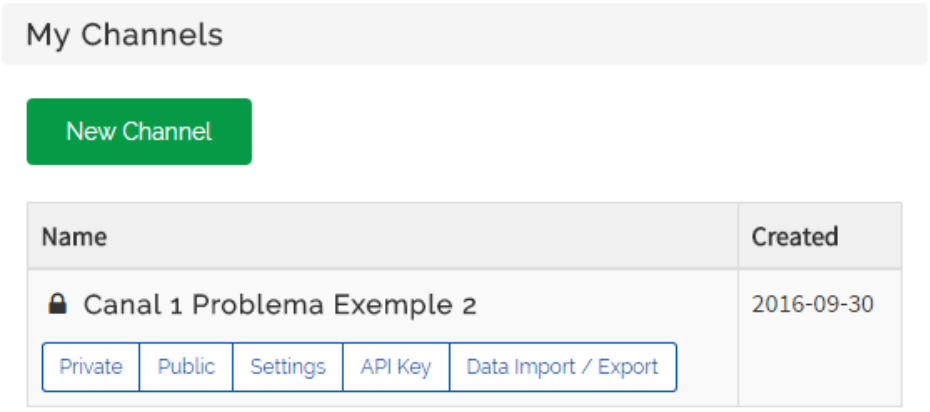

Figura 39: Llistat de canals. Font: https://thingspeak.com/.

Per crear els canals restants es clica de nou a "New Channel" i es realitza el mateix procediment, quedant els camps del segon canal:

Figura 40: Camps del canal 2 problema exemple 2: Font: Pròpia.

I els camps del tercer canal:

Figura 41: Camps del canal 3 problema exemple 2. Font: Própia.

Un cop creats els canals s'accedeix a la secció de "API Keys" de cadascun per tal de disposar de les claus d'escriptura:

- Canal 1: 017W8MISFWPRAK1T
- Canal 2: F0A1ZXS6M4C3JR8L
- Canal 3: ZQUX0C0R4G8IHDSM

5.3.5 Funció "Envia resultats"

Disposant ja de les claus d'escriptura, es procedeix a dissenyar el codi del botó "Envia resultats".

De nou en estructura és igual que en el problema exemple, però caldrà fer els canvis pertinents en les claus d'escriptura, noms de variables i ajustar el nombre de camps en el cas del canal 3:

```
function EnviaResultats_Callback(hObject, eventdata, handles)

if handles.avaluat == 1

    thingSpeakURL = 'http://api.thingspeak.com/';
    thingSpeakWriteURL = [thingSpeakURL 'update'];
```

– Es canvia la clau d'escriptura pel canal 1:

```
    writeApiKey1 = '017W8MISFWPRAK1T';

    fieldName11 = 'field1';
    fieldValue11 = handles.DNI;

    fieldName21 = 'field2';
    fieldValue21 = handles.nota;
```

– Es canvien el noms de les variables amb les respostes:

```
    fieldName31 = 'field3';
    fieldValue31 = handles.tfb1;

    fieldName41 = 'field4';
    fieldValue41 = handles.tfb2;

    fieldName51 = 'field5';
    fieldValue51 = handles.E;
```

- Es canvia la clau del canal 2:

```
writeApiKey2 = 'XZRRM71M72OY8FUZ';

fieldName12 = 'field1';
fieldValue12 = handles.ea;

fieldName22 = 'field2';
fieldValue22 = handles.eb;

fieldName32 = 'field3';
fieldValue32 = handles.ec;

fieldName42 = 'field4';
fieldValue42 = handles.ed;

fieldName52 = 'field5';
fieldValue52 = handles.ee;

fieldName62 = 'field6';
fieldValue62 = handles.ef;
```

- Es canvia la clau d'escriptura del canal 3:

```
writeApiKey3 = '6B6CDIRFL40Z0B2Z';
```

- Es canvien també els noms de les variables amb els paràmetres generats:

```
fieldName13 = 'field1';
fieldValue13 = handles.ptfb1;

fieldName23 = 'field2';
fieldValue23 = handles.ptfb2;

fieldName33 = 'field3';
fieldValue33 = handles.pE;
```

- I s'ajusten els inputs de les funcions "webwrite":

```
    response1 =
webwrite(thingSpeakWriteURL,'api_key',writeApiKey1,fieldName11,fieldVa
lue11,fieldName21,fieldValue21,fieldName31,fieldValue31,fieldName41,fi
eldValue41,fieldName51,fieldValue51);

    response2 =
webwrite(thingSpeakWriteURL,'api_key',writeApiKey2,fieldName12,fieldVa
lue12,fieldName22,fieldValue22,fieldName32,fieldValue32,fieldName42,fi
eldValue42,fieldName52,fieldValue52,fieldName62,fieldValue62);

    response3 =
webwrite(thingSpeakWriteURL,'api_key',writeApiKey3,fieldName13,fieldVa
lue13,fieldName23,fieldValue23,fieldName33,fieldValue33);


    if response1 == '0'

            set (handles.cEnvia, 'String', 'Error al penjar els
resultats, servidor ocupat. Intenta de nou en breu')

    elseif response2 == '0'

            set (handles.cEnvia, 'String', 'Error al penjar els
resultats, servidor ocupat. Intenta de nou en breu')

    elseif response3 == '0'

            set (handles.cEnvia, 'String', 'Error al penjar els
resultats, servidor ocupat. Intenta de nou en breu')
```

```
    else
            set (handles.cEnvia, 'String', 'Resultats penjats amb
èxit')

    end

else

    set (handles.cEnvia, 'String', 'Errors. És necessari haver-te
avaluat per penjar els resultats')

end
```

5.3.6 Fitxer distribuïble de codi ocult

Un cop dissenyat el codi de l'últim botó s'executa el programa i es comprova que tot funcioni sense errors.

En cas afirmatiu es procedeix a generar un fitxer de codi ocult per tal de poder distribuir-lo entre els usuaris.

Primerament cal unir els fitxers de la interfície i el codi en un de sol. Per fer-ho s'obre l'arxiu ."fig" que conté la interfície amb GUIDE. En la barra d'eines superior es clica a "File" i posteriorment a "Export" i es guarda l'arxiu generat (ProblemaExemple2_export).

Un cop generat l'arxiu s'oculta el seu codi mitjançant l'execució del següent comandament des de Matlab:

```
pcode 'ProblemaExemple2_export'
```

5.4 Visualitzacions

El següent pas és dissenyar les visualitzacions de ThingSpeak que permetran interpretar més fàcilment les dades que s'hi emmagatzemin.

Per tal d'accedir-hi, es clica a "Apps" a la part superior de la pantalla, on s'indica amb un requadre vermell en la figura a continuació:

Figura 42: Accés a aplicacions de ThingSpeak. Font: https://thingspeak.com/.

En la següent pàgina que es mostra es clica a l'apartat de visualitzacions:

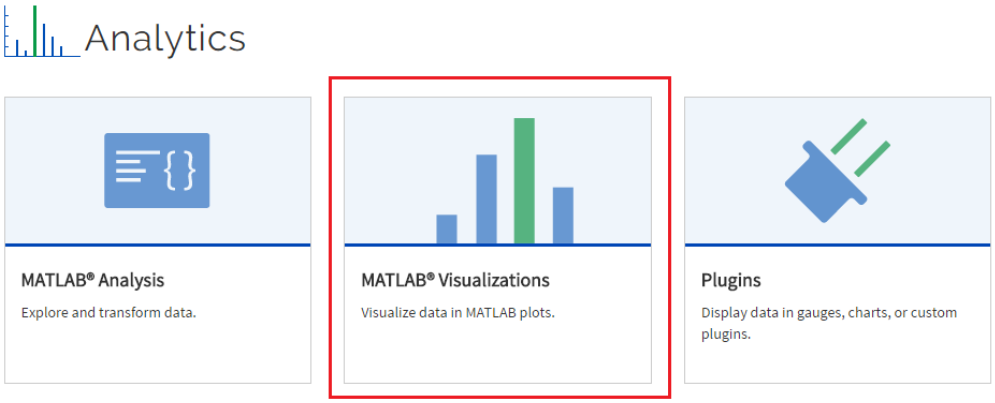

Figura 43: Accés a les visualitzacions. Font: https://thingspeak.com/.

Seguidament la pàgina que es mostra conté les visualitzacions creades. Com que és la primera que es crearà, només apareix l'opció de crear-ne una de nova. És clica sobre "New":

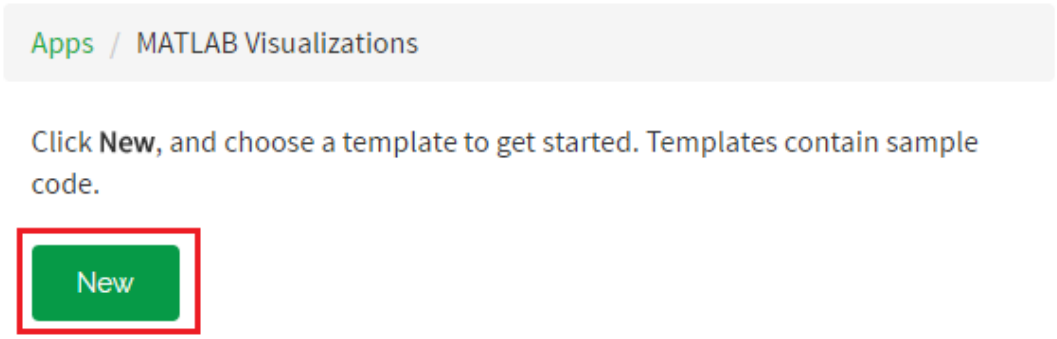

Figura 44: Panell de creació de visualitzacions. Font: https://thingspeak.com/.

Tot seguit es mostra un llistat de plantilles model per generar visualitzacions. Es selecciona la opció "Custom (no starter code)" i es clica sobre "Create" més avall en la mateixa pàgina:

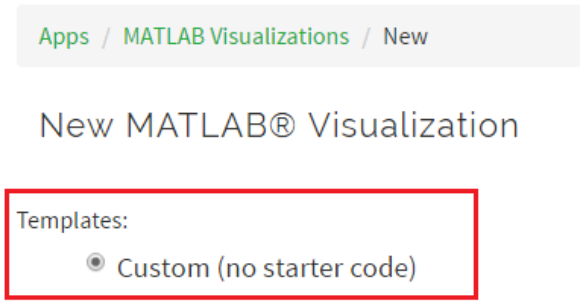

Figura 45: Opció de plantilla de codi per visualització. Font: https://thingspeak.com/.

La següent pàgina en mostrar-se és ja la de disseny de la visualització:

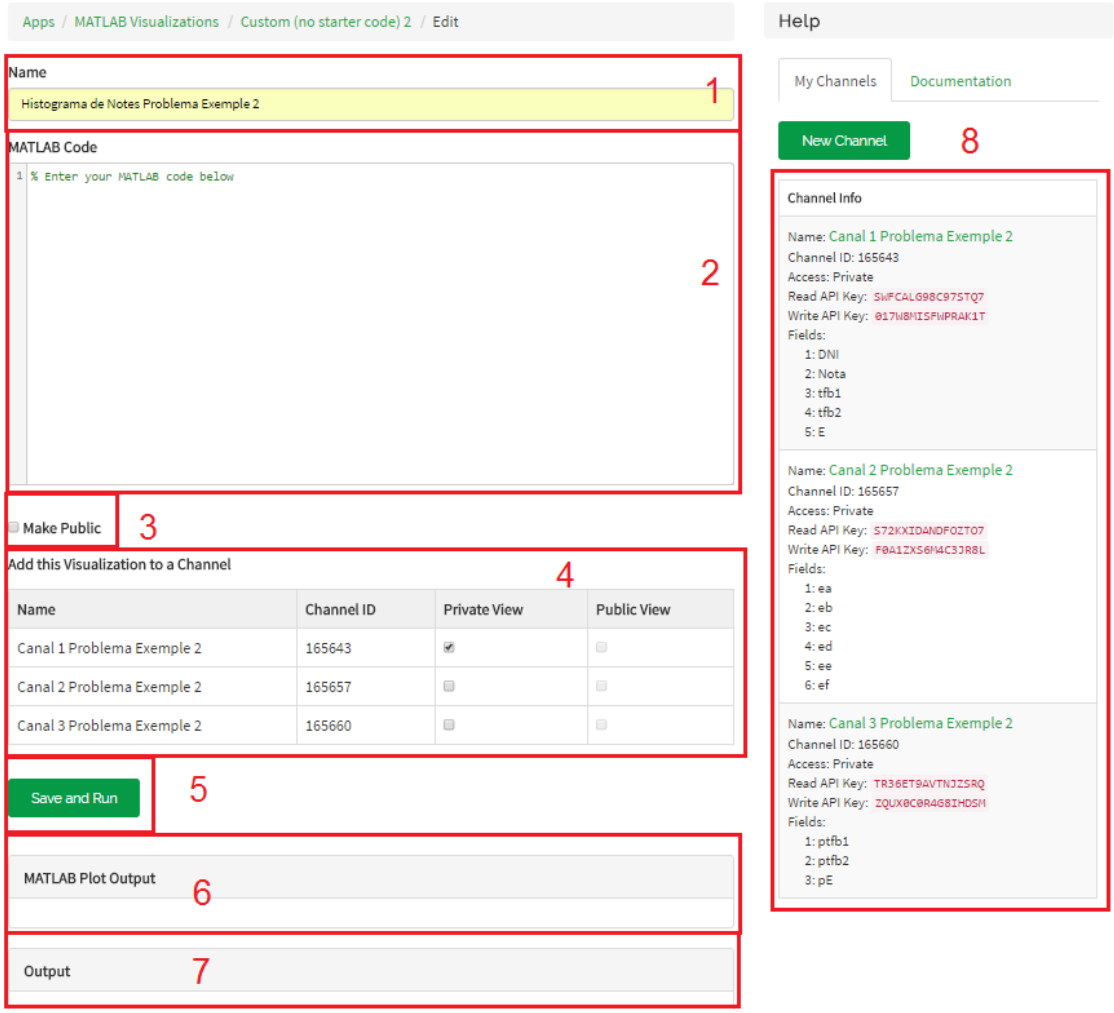

Figura 46: Pàgina de disseny de visualització. Font: https://thingspeak.com/.

On seguint la numeració de la figura 46:

1. Nom de la visualització.
2. Finestra de codi.
3. Pestanya per fer pública la visualització.
4. Panell d'assignació de visualitzacions. És marca la opció de veure la visualització en la vista privada en el canal 1. D'aquesta manera al accedir al canal es mostra la visualització.
5. Botó de guardar i executar codi.
6. Finestra de visualització generada.
7. Finestra de sortida.
8. Informació dels canals.

Per tal de generar l'histograma de notes, s'introdueix el mateix codi que l'utilitzat en el problema exemple en la finestra de codi, canviant únicament les dades del canal (ID i clau de lectura)

```
readChannelID = 165643;

readAPIKey = 'SWFCALG98C97STQ7';

notes = thingSpeakRead(readChannelID, 'Field', 2, 'NumPoints', 30, 'ReadKey', readAPIKey);


nota1=0;

nota2=0;

nota3=0;

nota4=0;


n=numel (notes);
```

```
for i = 1 : n
    if notes (i) == 10
        nota1 = nota1 + 1;

    elseif notes (i) == 6.67
        nota2 = nota2 + 1;

    elseif notes (i) == 3.33
        nota3 = nota3 + 1;

    elseif notes (i) == 0
        nota4 = nota4 + 1;
    end
end

hnotes = [nota1 nota2 nota3 nota4];
bar (hnotes)
xlabel('nota')
ylabel('freqüència')
```

Un cop introduït el codi es clica "Save and run" i es genera la visualització.

Per tal de generar l'histograma d'errors el procediment és el mateix, canviant també únicament en el codi la ID i la clau de lectura del canal corresponent:

```
readChannelID = 165657;

readAPIKey = 'S72KXIDANDFOZTO7';


errorsa = thingSpeakRead(readChannelID, 'Field', 1, 'NumPoints', 30,
'ReadKey', readAPIKey);

errorsb = thingSpeakRead(readChannelID, 'Field', 2, 'NumPoints', 30,
'ReadKey', readAPIKey);

errorsc = thingSpeakRead(readChannelID, 'Field', 3, 'NumPoints', 30,
'ReadKey', readAPIKey);

errorsd = thingSpeakRead(readChannelID, 'Field', 4, 'NumPoints', 30,
'ReadKey', readAPIKey);

errorse = thingSpeakRead(readChannelID, 'Field', 5, 'NumPoints', 30,
'ReadKey', readAPIKey);

errorsf = thingSpeakRead(readChannelID, 'Field', 6, 'NumPoints', 30,
'ReadKey', readAPIKey);


at=sum(errorsa);

bt=sum(errorsb);

ct=sum(errorsc);

dt=sum(errorsd);

et=sum(errorse);

ft=sum(errorsf);
```

```
errors = [at bt ct dt et ft ];

bar (errors);

xlabel('error')

ylabel('freqüència')
```

5.5 Anàlisi de dades: segona avaluació amb un error identificat

Per tal d'utilitzar l'eina d'anàlisi, es torna a la pàgina "Apps" i es clica l'apartat d'anàlisis de dades:

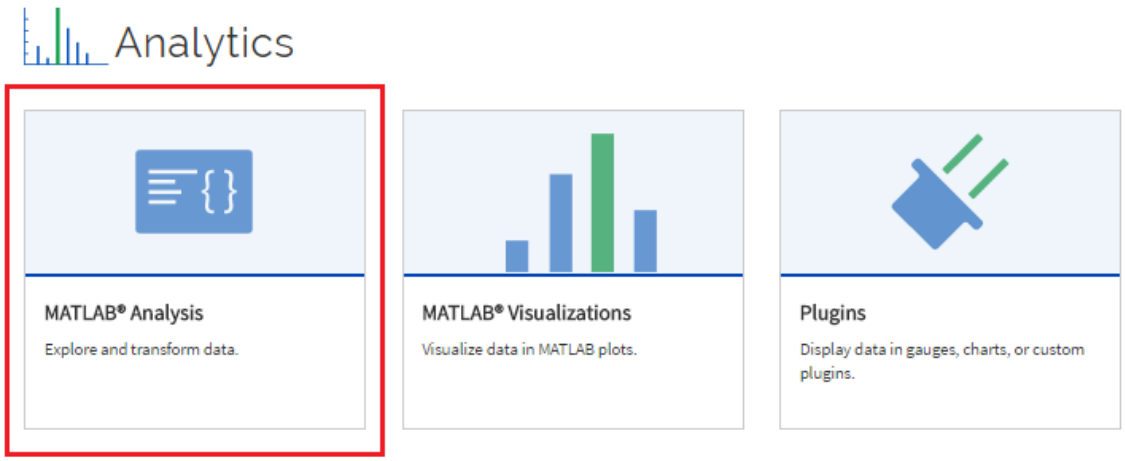

Figura 47: Accés a les anàlisis. Font: https://thingspeak.com/.

De la mateixa forma que amb les visualitzacions, es mostra una pantalla amb les anàlisis generades anteriorment. Es clica de nou sobre "New" i es mostra també una pantalla amb diferents plantilles d'anàlisis. Seguint el mateix procediment que amb les visualitzacions es selecciona la opció per generar una nova anàlisi sense codi inicial.

La pàgina que es mostra té un aspecte idèntic que la de les visualitzacions, en aquest cas però sense panell d'assignació de visualitzacions ni finestra de visualització generada:

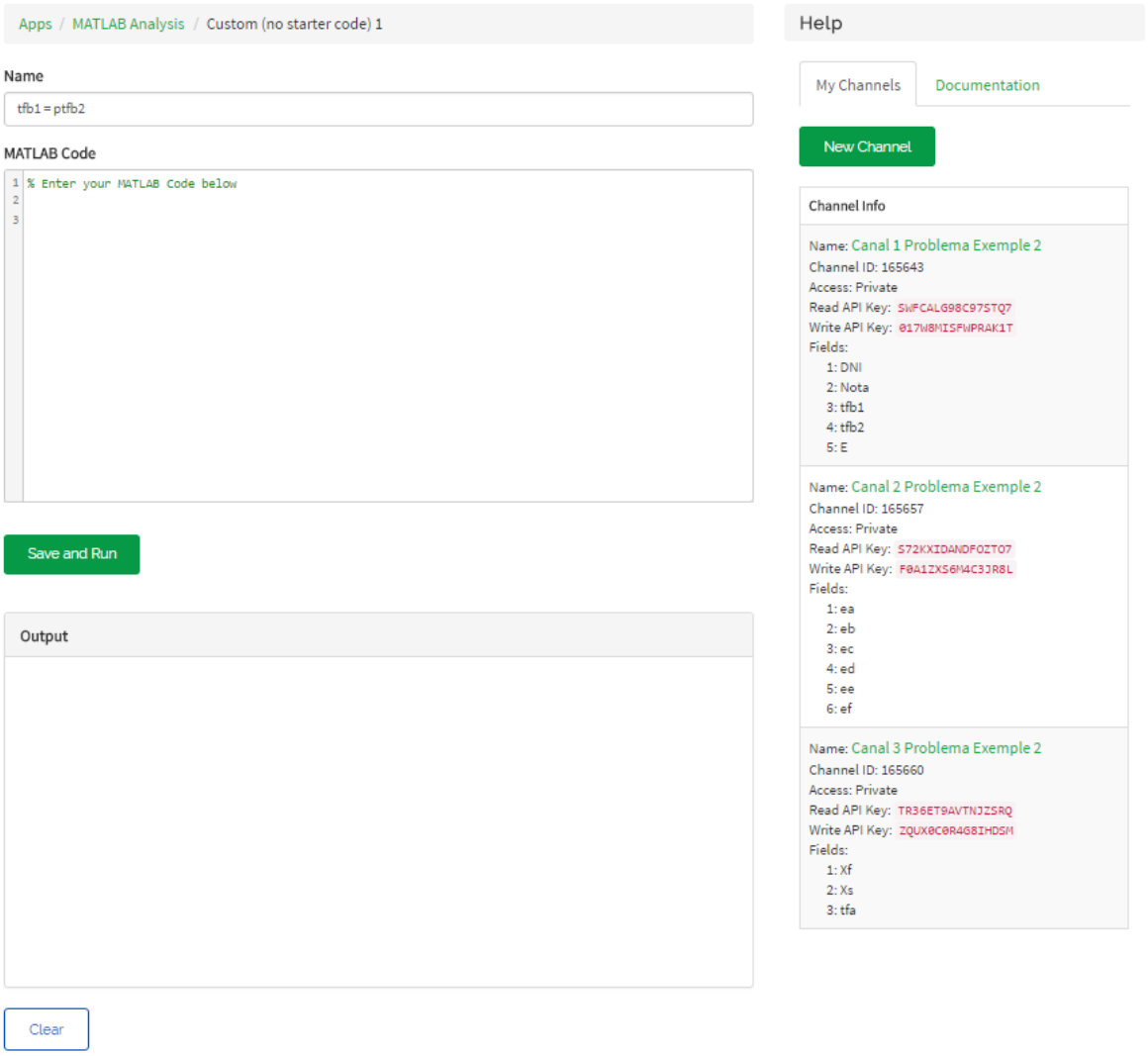

Figura 48: Pàgina de disseny d'anàlisi. Font: https://thingspeak.com/.

A partir d'aquest punt es procedeix a dissenyar el codi de la segona avaluació de les dades.

A mode d'exemple es vol comprovar si el valor donat a la primera qüestió correspon al valor que s'esperaria de la segona qüestió (tfb1 = ptfb2).

De nou el codi serà igual en estructura que el cas d'anàlisi del problema exemple, fent els canvis pertinents en les dades dels canals implicats, els noms de les variables i el procés de comparació:

- Es canvien IDs dels canals i les claus de lectura:

```
readChannelID1 = 165643;
readChannelID2 = 165660;
readAPIKey1 = 'SWFCALG98C97STQ7';
readAPIKey2 = 'TR36ET9AVTNJZSRQ';
```

- Es canvien els noms de les variables i s'ajusten els camps que es llegiran:

```
DNIs = thingSpeakRead(readChannelID1, 'Field', 1, 'NumPoints', 30,
'ReadKey', readAPIKey1);
ptfb2 = thingSpeakRead(readChannelID2, 'Field', 2, 'NumPoints', 30,
'ReadKey', readAPIKey2);
tfb1 = thingSpeakRead(readChannelID1, 'Field', 3, 'NumPoints', 30,
'ReadKey', readAPIKey1);

n=numel (DNIs);
epsilon = 0.05;
DNI= 0;
errorg = 0;


for i = 1 : n

    eg = 0;
```

- Es canvia el nom de la variable que conté els valors que prendrà la resposta de l'usuari si s'ha comès l'error que s'analitza (és a dir, el valor de la segona qüestió calculada pel programa)

```
ptfb1e = ptfb2(i);
```

– S'ajusten les variables en el càlcul de la diferència absoluta entre el valor de l'error i la resposta introduïda per l'usuari:

```
difptfb1e = abs(ptfb1e-tfb1(i));
```

– I s'ajusten també les variables en el condicional:

```
if difptfb1e< epsilon*ptfb1e

  eg =  1;


if DNI == 0

    DNI = DNIs (i);
    errorg = eg;

  else

    DNI = [DNI;DNIs(i)];
    errorg = [errorg; eg];

  end

else

  if DNI == 0

    DNI = DNIs (i);
    errorg = eg;
```

```
    else

      DNI = [DNI;DNIs(i)];
      errorg = [errorg; eg];

    end
  end

end



format long

table (DNI, errorg)
```

Abans d'introduir el codi es generen dues entrades amb el l'arxiu del problema on en la primera no es comet l'error però si en la segona.

Un cop generades les entrades s'introdueix el codi i s'executa mitjançant el botó "Save and run". En la finestra de sortida es mostra la taula:

DNI	errorg
39395908	0
39395908	1

Figura 49: Taula del resultat de l'anàlisi problema exemple 2. Font: Pròpia.

On es pot observar que l'anàlisi de l'error dona negatiu per la primera entrada i positiu per la segona.

5.6 Exportació de dades

Per treballar les dades en un entorn diferent del de ThingSpeak, es poden exportar les dades en un arxiu CSV. Per fer-ho es clica la pestanya "Data Import / Export" situada a la capçalera de la pàgina principal del canal corresponent i posteriorment en la pestanya de descarrega en l'apartat d'exportar:

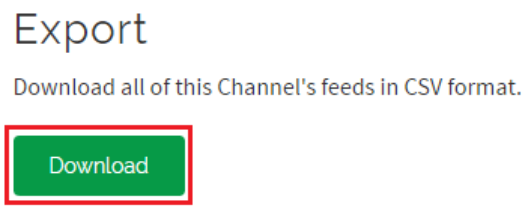

Figura 50: Casella d'exportació de dades. Font: https://thingspeak.com/.

L'arxiu CSV descarregat pot obrir-se en un Excel i presenta les dades compactes en una columna on cada fila correspon a una entrada:

	A	B	C	D	E
1	created_at,entry_id,field1,field2,field3,field4,field5				
2	2016-09-30 21:20:25 UTC,1,39395908,0,"","",""				
3	2016-10-01 14:48:38 UTC,2,39395908,6.67,74.5125,74.5125,2				
4	2016-10-01 14:55:32 UTC,3,39395908,10,39.714,74.5125,2				
5	2016-10-01 14:55:49 UTC,4,39395908,0,0,0,0				

Figura 51: Aspecte inicial de les dades descarregades. Font: https://thingspeak.com/.

Per tal de mostrar les dades en diferents columnes per ser compreses més fàcilment cal seleccionar les dades i clicar "Texto en columnas" en la pestanya "Datos":

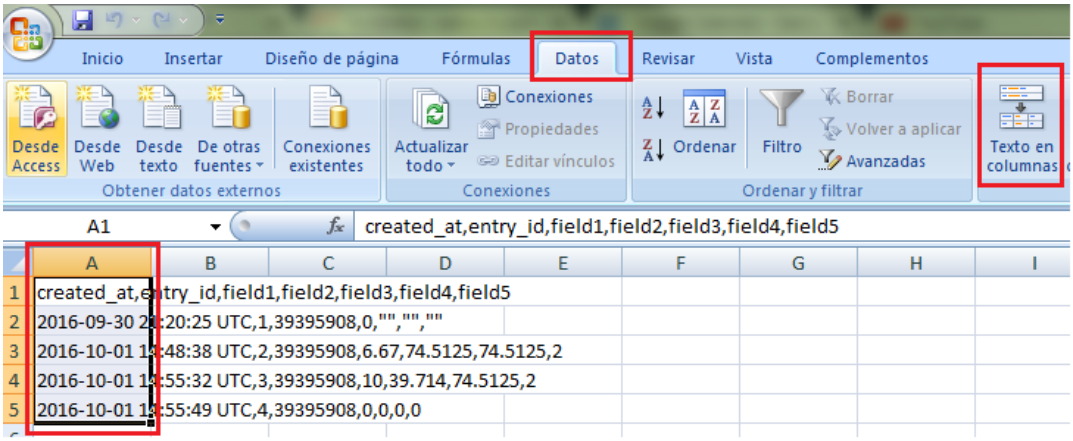

Figura 52: Adequació de format de les dades descarregades (1). Font: Pròpia.

En la següent finestra que s'obre es selecciona "Delimitados" i es clica "Siguiente":

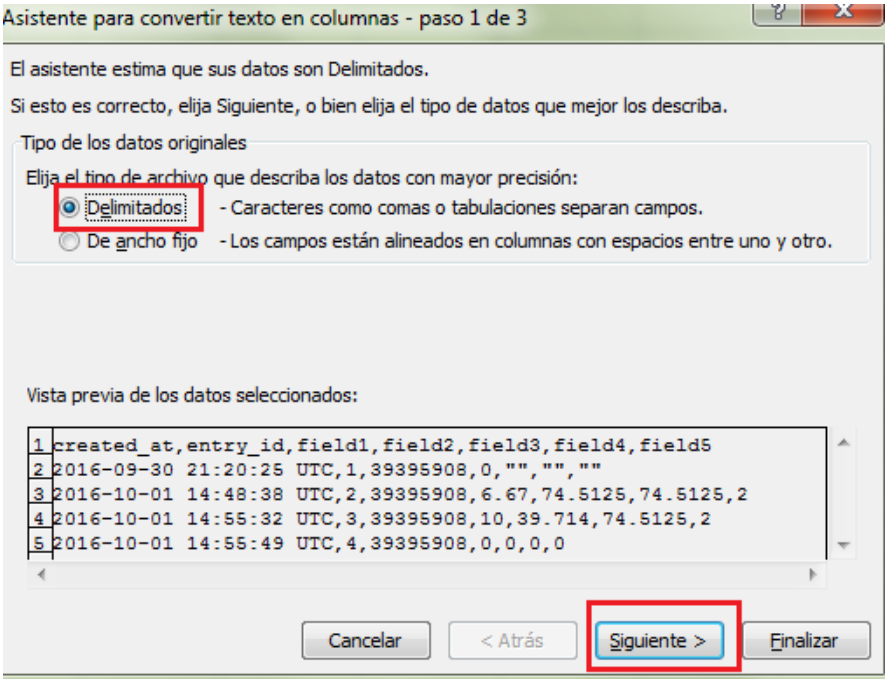

Figura 53: Adequació de format de les dades descarregades (2). Font: Pròpia.

A continuació s'indica que els separadors són les comes i es clica "Siguiente":

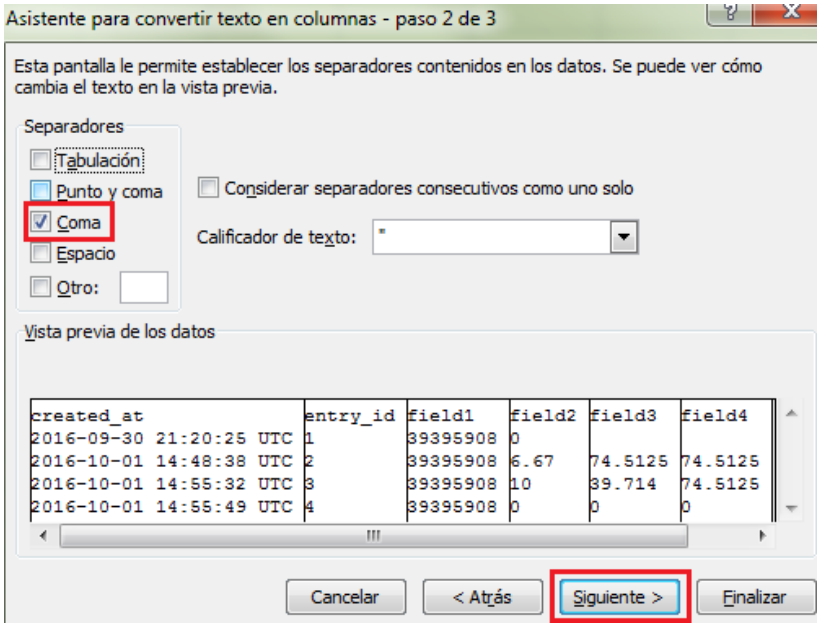

Figura 54: Adequació de format de les dades descarregades (3). Font: Pròpia.

Finalment es selecciona el format de dades en les columnes general i es clica "Finalizar":

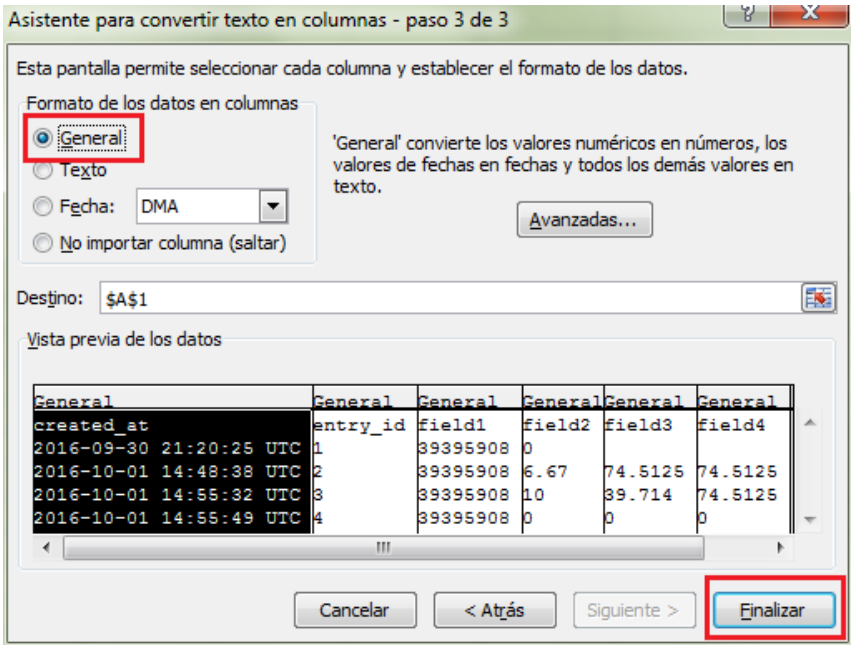

Figura 55: Adequació de format de les dades descarregades (4). Font: Pròpia.

Un cop adequat el format les dades presenten el següent aspecte:

A	B	C	D	E	F	G
created_at	entry_id	field1	field2	field3	field4	field5
2016-09-30 21:20:25 UTC	1	39395908	0			
2016-10-01 14:48:38 UTC	2	39395908	6.67	745.125	745.125	2
2016-10-01 14:55:32 UTC	3	39395908	10	39.714	745.125	2
2016-10-01 14:55:49 UTC	4	39395908	0	0	0	0

Figura 56: Format final de les dades descarregades. Font: Pròpia.

6. Tipologia de problemes

L'extensa gama de problemes que cobreix Matlab fa que sigui un llenguatge difícil de comprendre i manejar en la seva plenitud. Això no vol dir que sigui inabordable, el coneixement base que permet començar a treballar és molt senzill (Domíınguez Báguena, V., Rapún Banzo, M. L. (2006). Matlab en cinco lecciones de Numérico, 1-2.).

No obstant, l'elevat número de comandaments provoca que en ocasions existeixin problemes no tant sols per trobar els comandaments adequats, sinó també per fer-se una idea de les possibilitats que ofereix exactament Matlab en un problema o tasca en particular.

En conseqüència, més que preguntar-se quines possibilitats ofereix Matlab, cal preguntar-se quin és l'objectiu que es vol assolir i investigar una forma per aproximar-s'hi.

El programari desenvolupat es plantejava en un inici enfocat a la resolució de problemes d'enginyeria química, però pot ser utilitzat en problemes de tota mena mentre les respostes siguin valors numèrics comparables o opcions que puguin ser numerades.

De fet, analitzant els beneficis d'utilitzar el programa, les assignatures que en sortirien més beneficiades són les que tenen grups d'alumnes més nombrosos, ja que és en aquestes on la comunicació entre docents i alumnes és més complicada i el temps que el docent ha d'invertir en tasques mecàniques de correcció és major, malgastant recursos que poden aprofitar-se en tasques més creatives i d'adaptació al nivell i necessitats actuals dels alumnes.

7. Conclusions

L'objectiu principal que es plantejava en l'inici del projecte ha pogut ser complert amb èxit, aconseguint desenvolupar un programari capaç d'assolir les funcions de disseny, resolució i correcció automàtica de problemes que es desitjava aconseguir, juntament amb la característica única de processar les dades emmagatzemades que permet al docent disposar de recursos per interpretar i analitzar les dades així com d'automatitzar la cerca d'errors comuns entre els usuaris.

No obstant, el programa desenvolupat no ha estat testat amb grups d'alumnes ni focalitzat a cap cas concret més enllà dels problemes exemple plantejats pel seu desenvolupament i per a l'elaboració de la guia per dissenyar problemes utilitzant el programari.

El resultat final és doncs un programari versàtil que ofereix moltes opcions i que tot i que s'hagi desenvolupat a partir d'objectius basats en l'entorn de la docència, la programació mitjançant GUIDE permet transportar les aplicacions desenvolupades a l'àmbit industrial per tal de permetre a l'enginyer desenvolupar i comercialitzar programes per a casos específics a nivell de disseny, operació o optimització.

Seria doncs el següent pas analitzar més profundament les possibles aplicacions que es puguin derivar del programari en diferents àmbits i estudiar quines assignatures sortirien més beneficiades d'incorporar el programari desenvolupat per tal d'obtenir dades en la seva implementació que en permetin jutjar el seu desenvolupament i recollir opinions dels usuaris.

8. Futures línies de treball

El programari desenvolupat és la base per construir multitud de problemes i aplicacions amb finalitats diverses.

Essent una base com és, els següents passos per aplicar el programari passen per analitzar i concretar en quins casos el programari pot resultar útil.

Seguint aquesta línia en l'àmbit docent, es podria començar a dissenyar un repertori de problemes amb l'objectiu de ser proposats a l'alumnat i obtenir dades i opinions de la seva implementació per tal d'analitzar els resultats obtinguts en el seu ús i valorar el seu desenvolupament més enllà del marc teòric.

9. Bibliografia

Mathworks. Creación de apps con interfaces gráficas de usuario en MATLAB. [en línia]. [consulta: 30-01-2016]. Disponible a: de http://es.mathworks.com/discovery/matlab-gui.html

Mathworks. Initialize UI Components in GUIDE Apps. Opening Function . [en línia]. [consulta: 30-01-2016]. Disponible a: http://es.mathworks.com/help/matlab/creating_guis/initializing-a-guide-gui.html

Mathworks. Write Callbacks in GUIDE. Callbacks for Different User Actions. [en línia]. [consulta: 30-01-2016]. Disponible a: http://es.mathworks.com/help/matlab/creating_guis/write-callbacks-using-the-guide-workflow.html#f10-999868

Mathworks. guidata. Store or retrieve UI data. [en línia]. [consulta: 03-02-2016]. Disponible a: http://es.mathworks.com/help/matlab/ref/guidata.html?refresh=true

Mathworks. get (COM). Get property value from interface, or display properties. [en línia]. [consulta: 03-02-2016]. Disponible a: http://es.mathworks.com/help/matlab/ref/com.get.html

Mathworks. set. Set graphics object properties. [en línia]. [consulta: 03-02-2016]. Disponible a: http://es.mathworks.com/help/matlab/ref/set.html?refresh=true

Mathworks. str2num. Convert character array to numeric array. [en línia]. [consulta: 03-02-2016]. Disponible a: https://es.mathworks.com/help/matlab/ref/str2num.html

Mathworks. size. Array size. [en línia]. [consulta: 03-02-2016]. Disponible a: http://es.mathworks.com/help/matlab/ref/size.html

Mathworks. isempty. Determine whether array is empty. [en línia]. [consulta: 14-02-2016]. Disponible a: https://es.mathworks.com/help/matlab/ref/isempty.html

Mathworks. abs. Absolute value and complex magnitude. [en línia]. [consulta: 14-02-2016]. Disponible a: https://es.mathworks.com/help/matlab/ref/abs.html

Mathworks. round. Round to nearest decimal or integer. [en línia]. [consulta: 14-02-2016]. Disponible a: http://es.mathworks.com/help/matlab/ref/round.html

Mathworks. double. Convert symbolic values to MATLAB double precision. [en línia]. [consulta: 14-02-2016]. Disponible a: de https://es.mathworks.com/help/symbolic/double.html

Mathworks. log. Natural logarithm. [en línia]. [consulta: 12-03-2016]. Disponible a: http://es.mathworks.com/help/matlab/ref/log.html

Mathworks. Solve, Equations and systems solver. [en línia]. [consulta: 12-03-2016]. Disponible a: http://es.mathworks.com/help/symbolic/solve.html

Mathworks. if, elseif, else. Execute statements if condition is true. [en línia]. [consulta: 12-03-2016]. Disponible a: http://es.mathworks.com/help/matlab/ref/if.html

Mathworks. Getting Started with ThingSpeak. Introduction to ThingSpeak™ and overview of its basic workflow. [en línia]. [consulta: 04-04-2016]. Disponible a: https://es.mathworks.com/help/thingspeak/getting-started-with-thingspeak.html

Mathworks. Channel Configurations. ThingSpeak Channel Access. [en línia]. [consulta: 04-04-2016]. Disponible a: http://es.mathworks.com/help/thingspeak/channel-settings.html

Mathworks. webwrite. Write data to RESTful web service. [en línia]. [consulta: 07-05-2016]. Disponible a: http://es.mathworks.com/help/matlab/ref/webwrite.html

Mathworks. thingSpeakRead. Read data stored in ThingSpeak channel. [en línia]. [consulta: 07-05-2016]. Disponible a: de http://es.mathworks.com/help/thingspeak/thingspeakread.html

Mathworks. MATLAB Analysis and Visualization. Explore and transform data, visualize data in MATLAB® plots. [en línia]. [consulta: 08-06-2016]. Disponible a: http://es.mathworks.com/help/thingspeak/matlab-analysis-and-visualization.html

Mathworks. ThingSpeak. Analyze Your Data. [en línia]. [consulta: : 08-06-2016]. Disponible a: https://es.mathworks.com/help/thingspeak/analyze-your-data.html

Mathworks. Create a Chart. [en línia]. [consulta: : 08-06-2016]. Disponible a: https://es.mathworks.com/help/thingspeak/create-a-chart.html

Mathworks. for. for loop to repeat specified number of times. [en línia]. [consulta: : 08-06-2016]. Disponible a: https://es.mathworks.com/help/matlab/ref/for.html

Mathworks. bar. Bar graph. [en línia]. [consulta: : 08-06-2016]. Disponible a: https://es.mathworks.com/help/matlab/ref/bar.html

Mathworks. table. Create table from workspace variables. [en línia]. [consulta: : 08-06-2016]. Disponible a: http://es.mathworks.com/help/matlab/ref/table.html

Mathworks. numel. Number of array elements. [en línia]. [consulta: : 08-06-2016]. Disponible a: http://es.mathworks.com/help/matlab/ref/numel.html

Mathworks. sum. Sum of array elements. [en línia]. [consulta: : 08-06-2016]. Disponible a: http://es.mathworks.com/help/matlab/ref/sum.html

Mathworks. pcode. Create protected function file. [en línia]. [consulta: : 12-08-2016]. Disponible a: http://es.mathworks.com/help/matlab/ref/pcode.html

Tiching. Tiching. El Blog de Educación y TIC. Neus Sanmartí: «Sólo aprende quien se autoevalúa». [en línia]. [consulta: 10-09-2016]. Disponible a: http://blog.tiching.com/neus-sanmarti-solo-aprende-quien-se-autoevalua/

Siegfried, D. S. Futurehumanevolution. Cognitive Psychology: Problem Solving. [en línia]. [consulta: 10-09-2016]. Disponible a: http://futurehumanevolution.com/cognitive-psychology-and-problem-solving

Antoni, F. R. (2004). Líneas maestras en el Aprendizaje por Problemas. *Revista interuniversitaria de formación del profesorado*, *18*(49), 79-96.

Equipo Docente en ABP. Facultad de Psicología. Universidad de Murcia (2011). El proceso de evaluación en la metodología de Aprendizaje Basado en Problemas.

Fernández, L., Fernández-gao, C., Arenas, G., Peña, M. De, Fernández, C. L., & Gómez-estern, F. (2014). Evaluación continua con Goodle-GMS a más de 800 alumnos en Ingeniería Química. doi:10.7203/attic.13.3855

Molina, J. A., García, A., Pedraz, A., & Antón, M. V. (2003). Aprendizaje basado en problemas : una alternativa al método tradicional. *Revista de la Red Estatatla de Docencia Universitaria*, *3*, 79-85.

Domíınguez Báguena, V., Rapún Banzo, M. L. (2006). Matlab en cinco lecciones de Numérico, 1-2.

Doménech, P. M. (2014). *II Congreso de Innovación Docente en Ingeniería Química,* 52

Muñoz, D., Peña, D., & Gómez-estern, F. Tutorial de Doctus.

Muñoz, D., Peña, D., & Gómez-estern, F. Doctus Diseño de ejercicios y casos prácticos Índice, 1-56.

Gómez-estern, F. Tutorial de Doctus Gestión de suscripciones.

Gómez-estern, F.Tutorial de Doctus exacta.

Gómez-estern, F. Evaluación Automática Mediante Doctus.